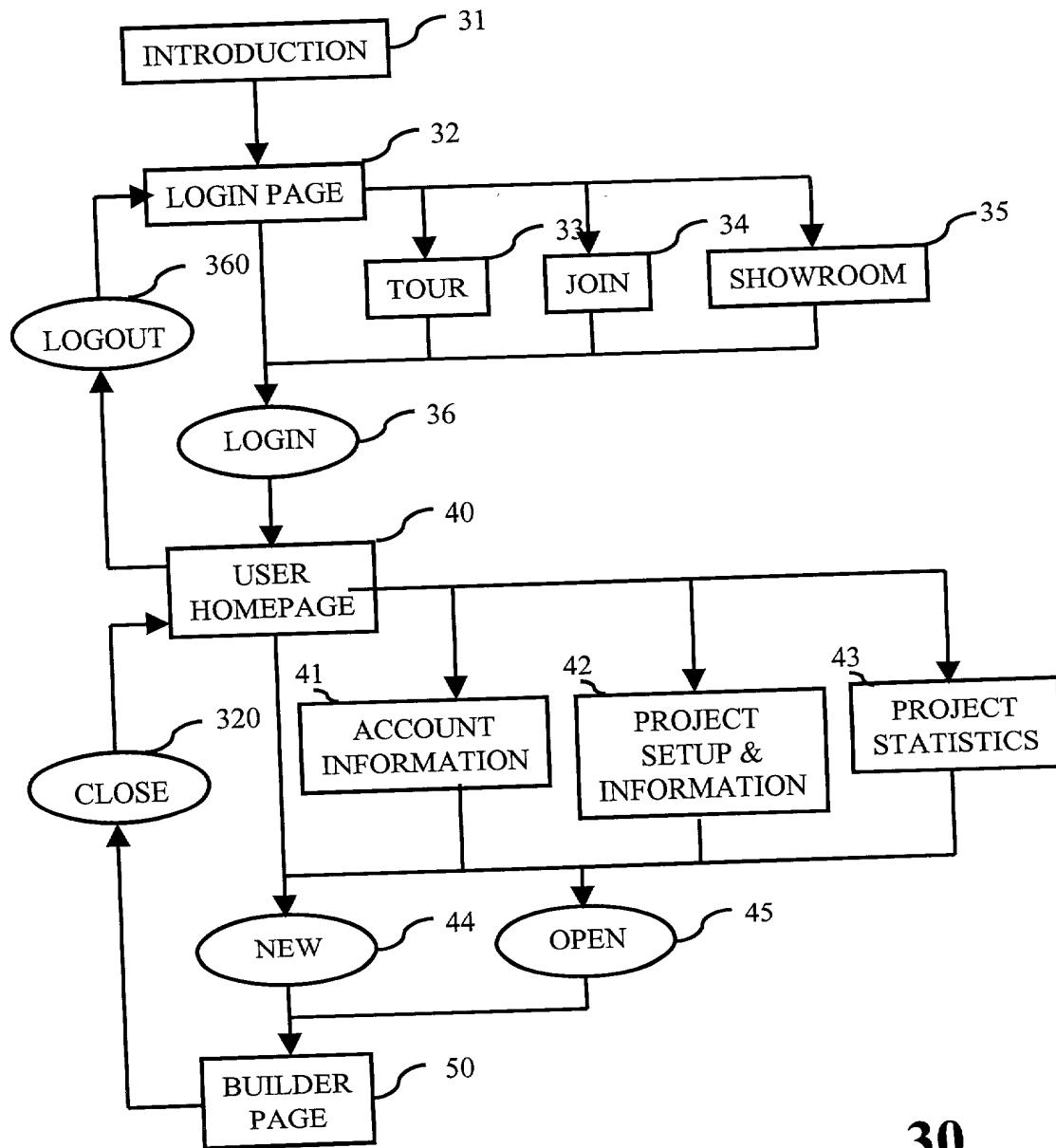


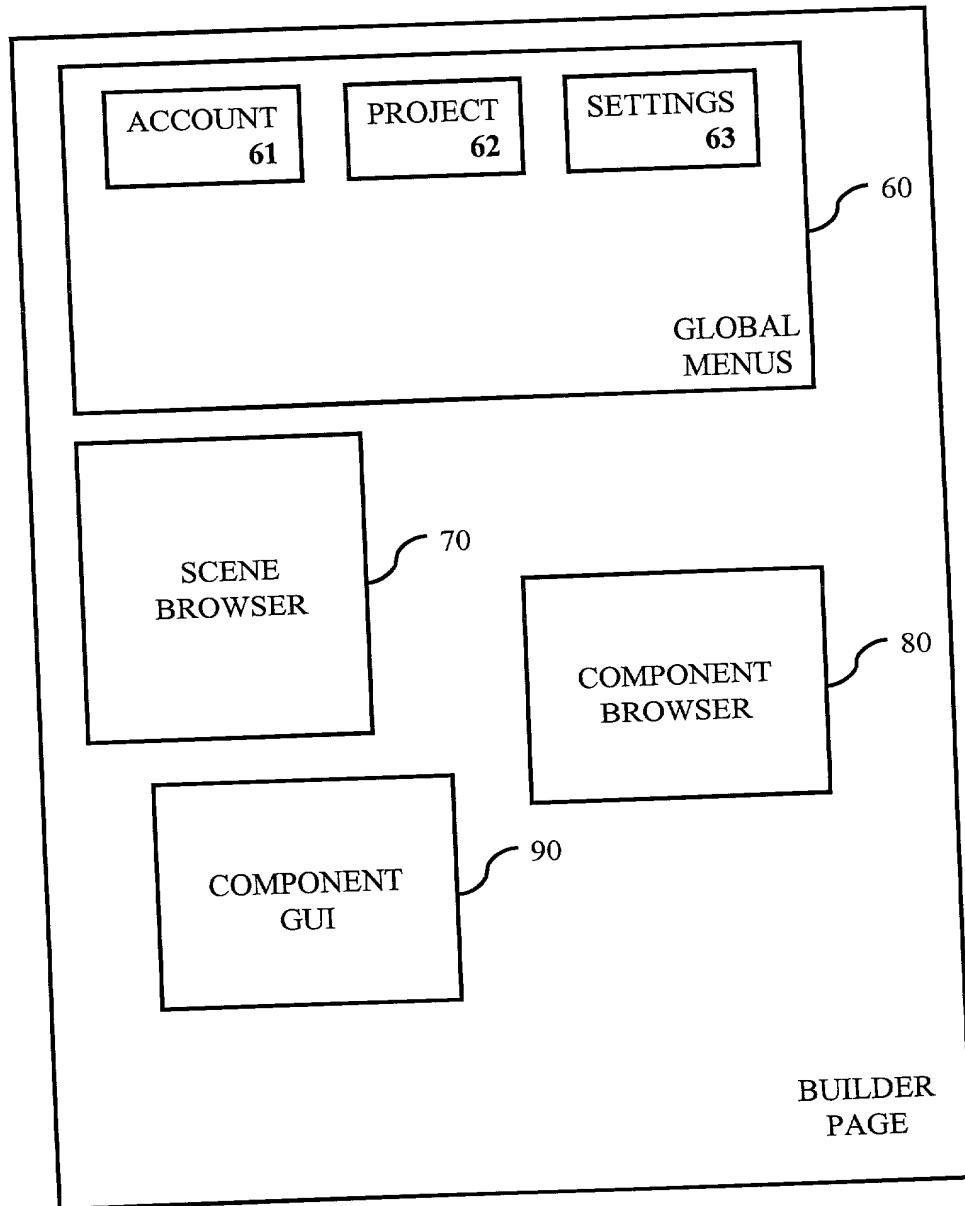
10

Fig. 1

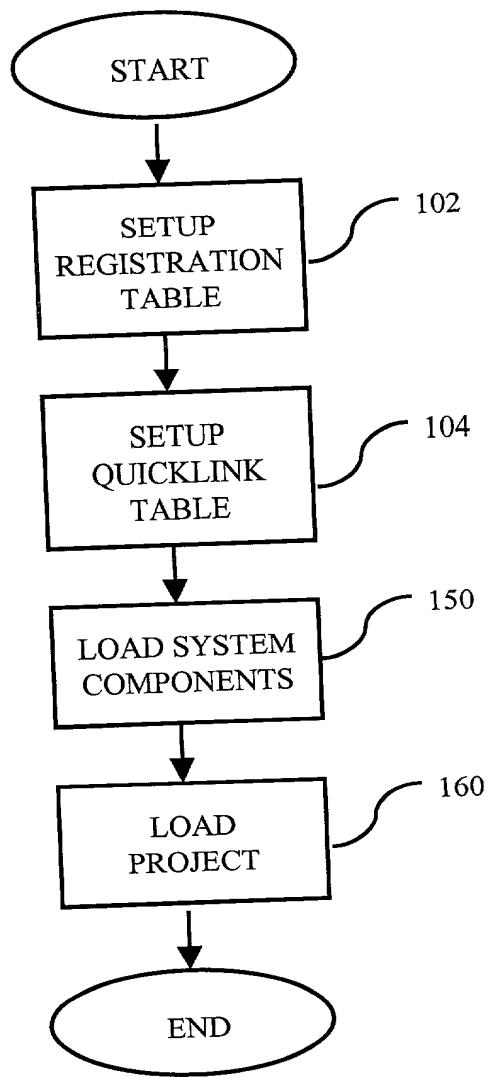


30

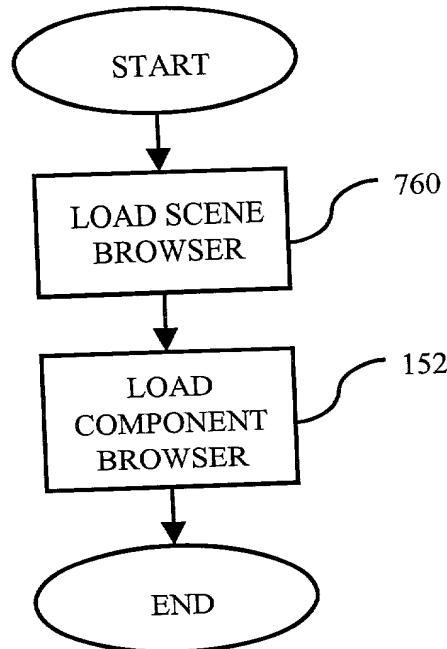
Fig. 2



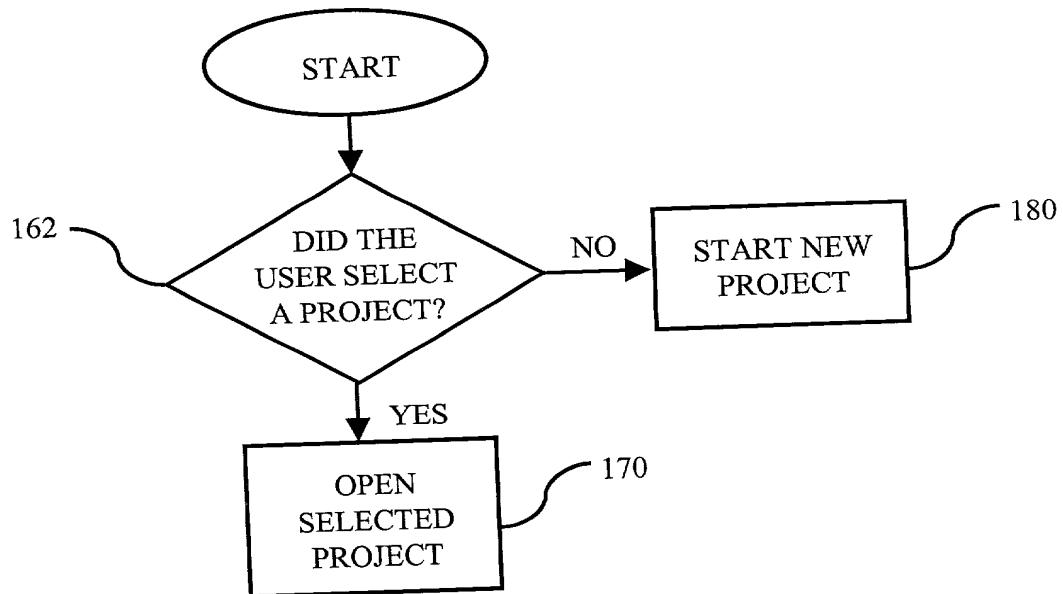
50
Fig. 3



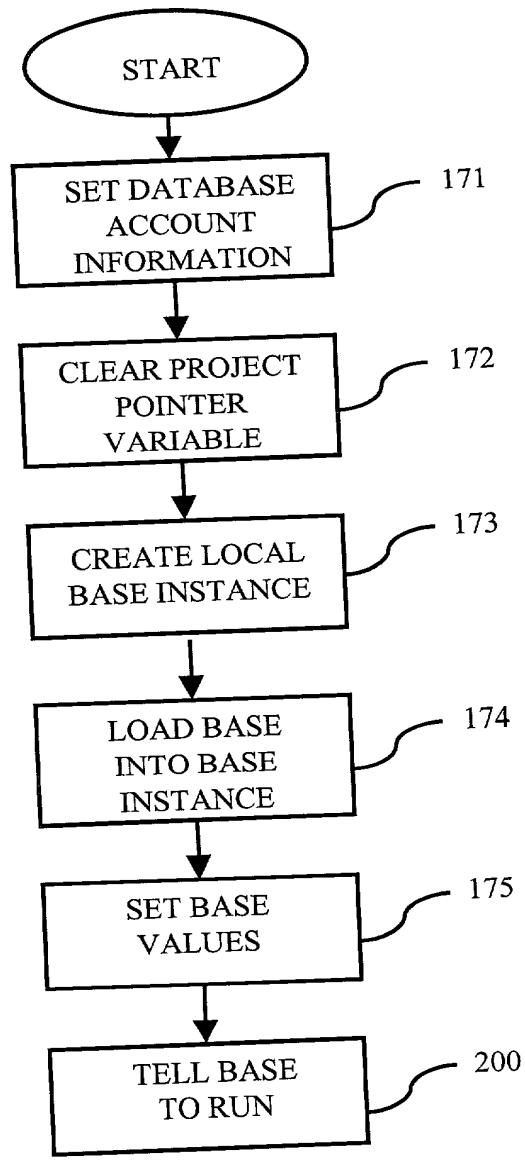
100
Fig. 4

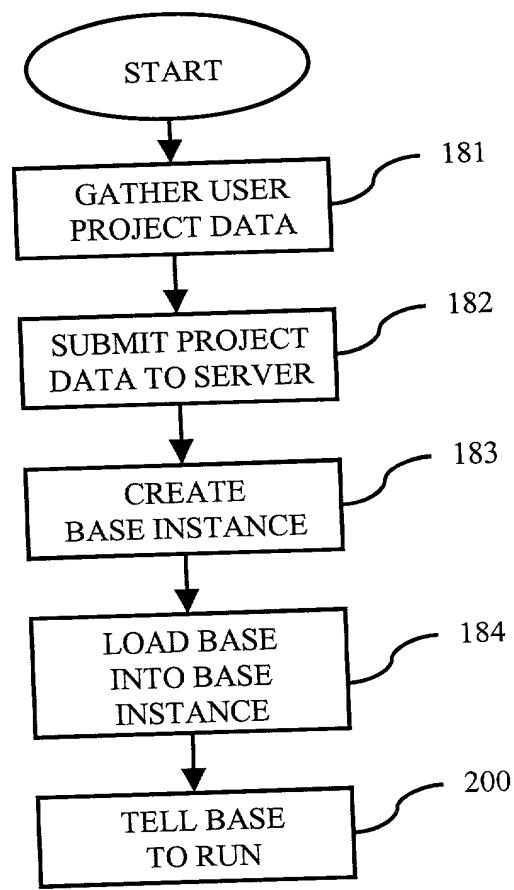


150
Fig. 5

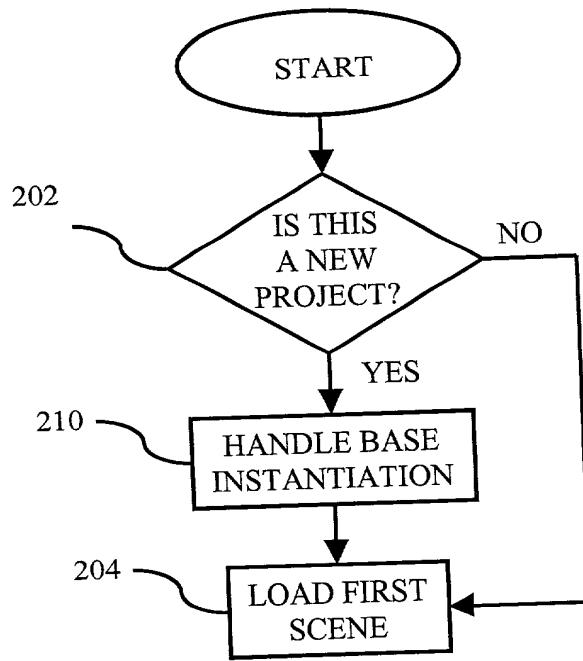


160
Fig. 6

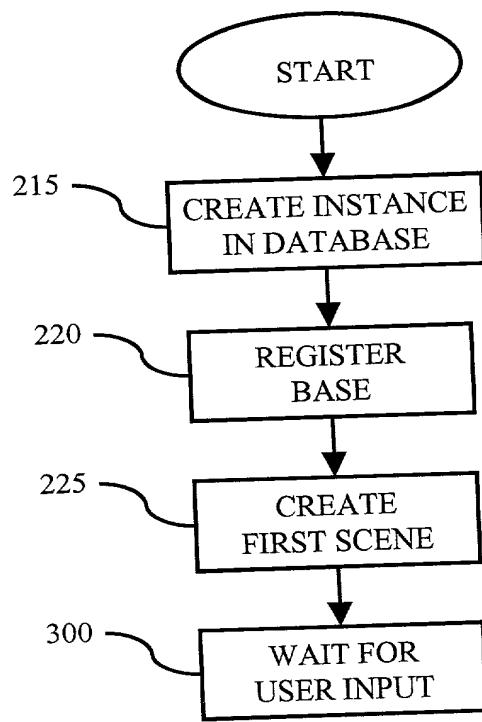




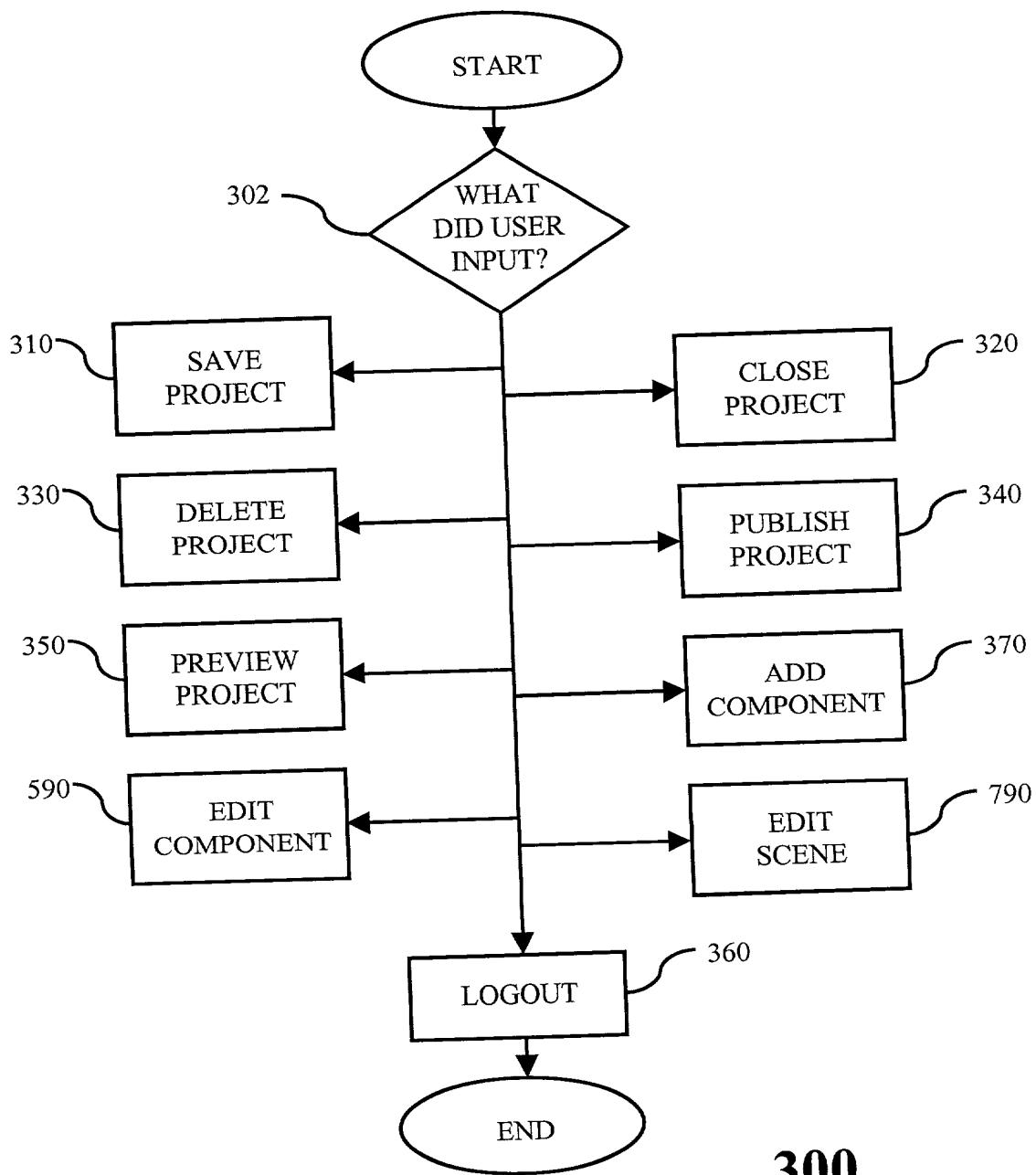
180
Fig. 8



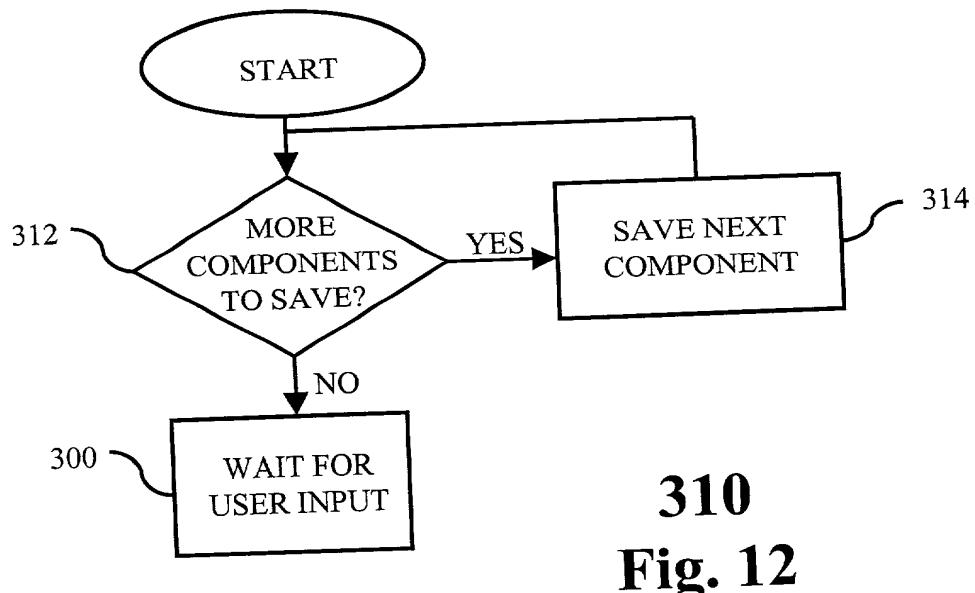
200
Fig. 9



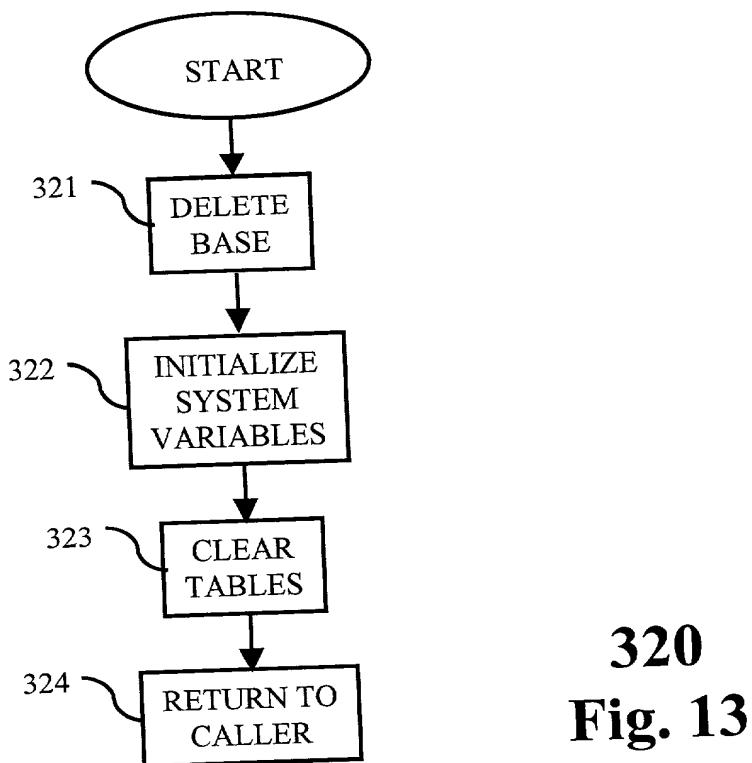
210
Fig. 10



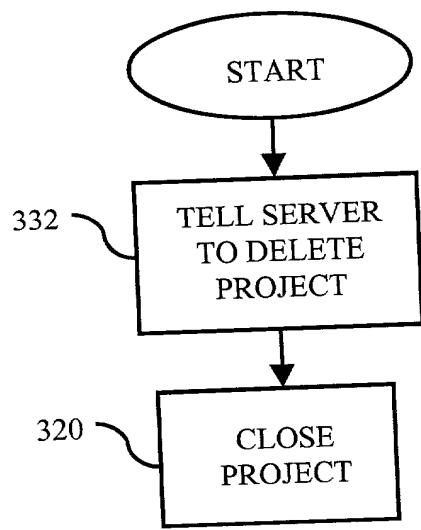
300
Fig. 11



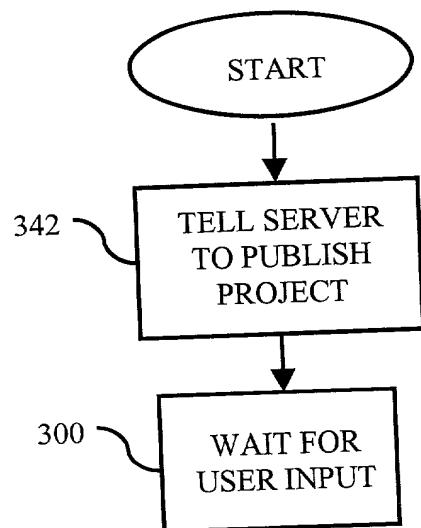
310
Fig. 12



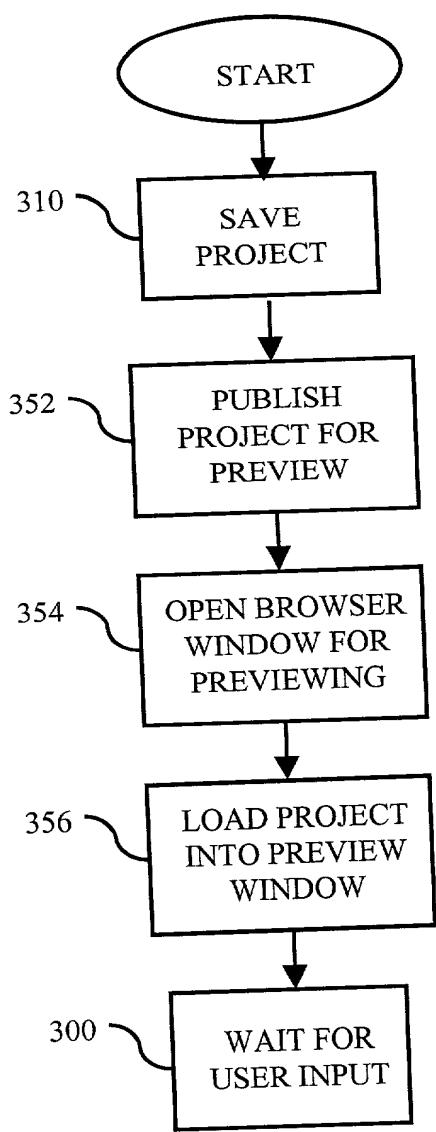
320
Fig. 13



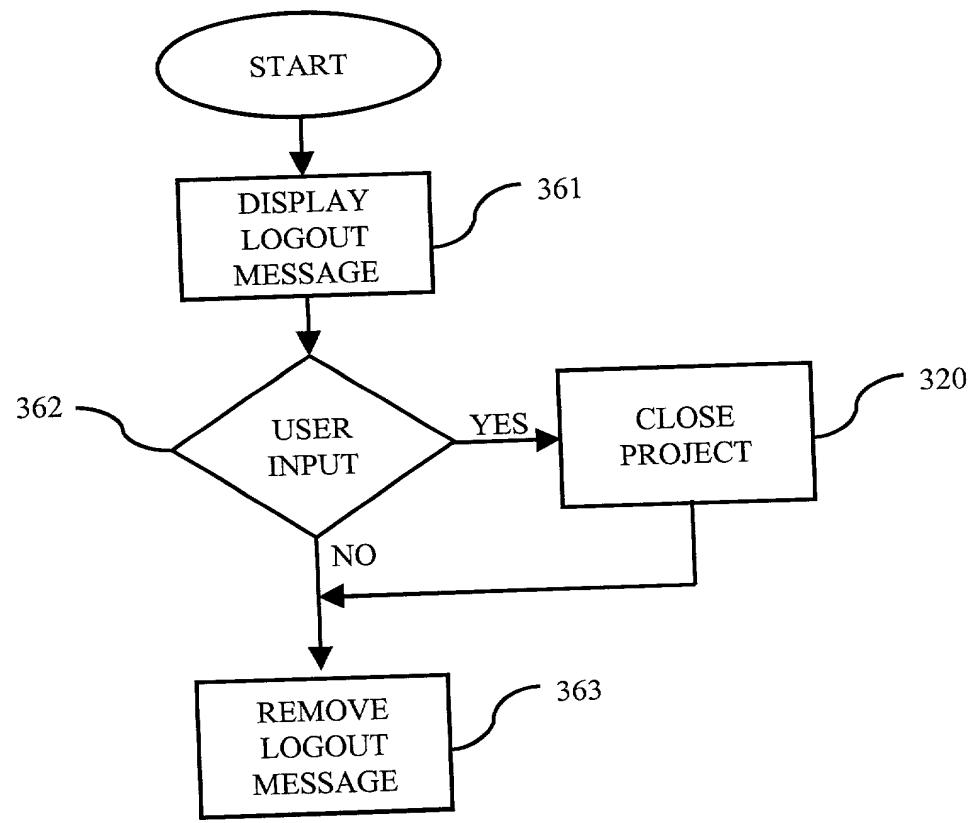
330
Fig. 14



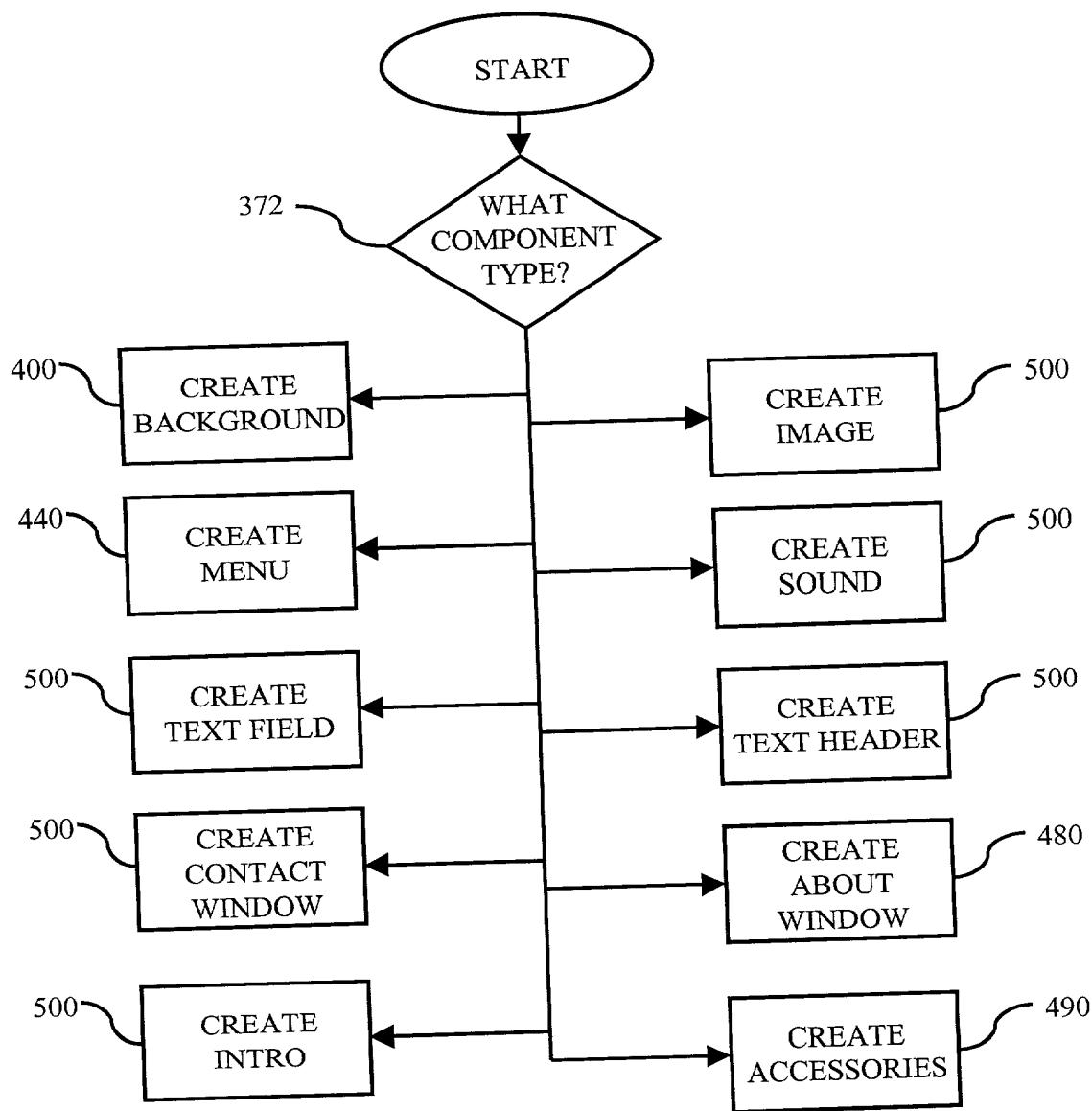
340
Fig. 15



350
Fig. 16

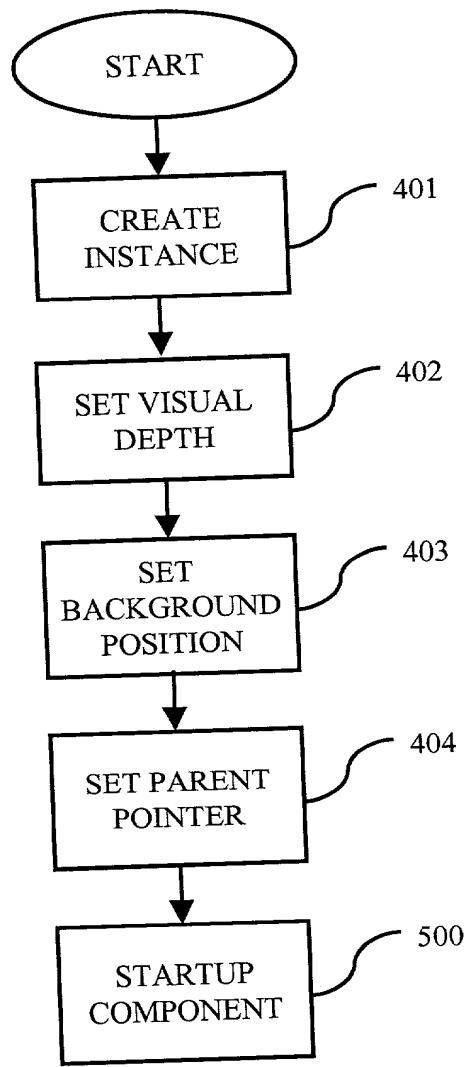


360
Fig. 17

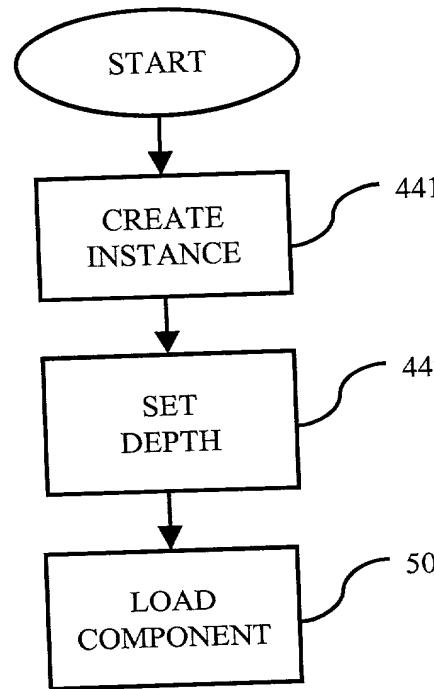


370

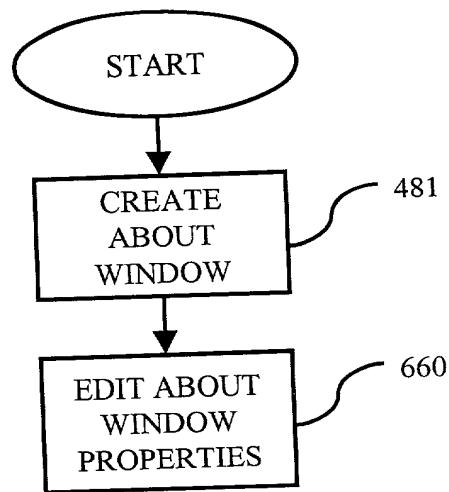
Fig. 18



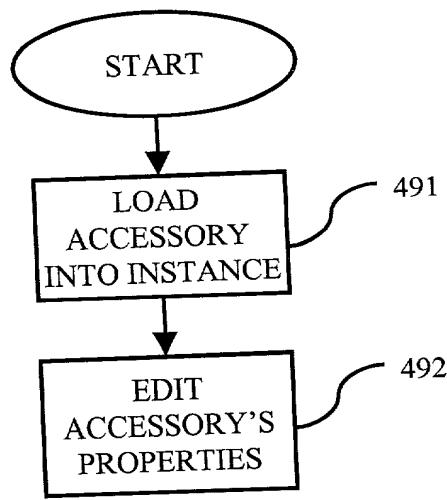
400
Fig. 19



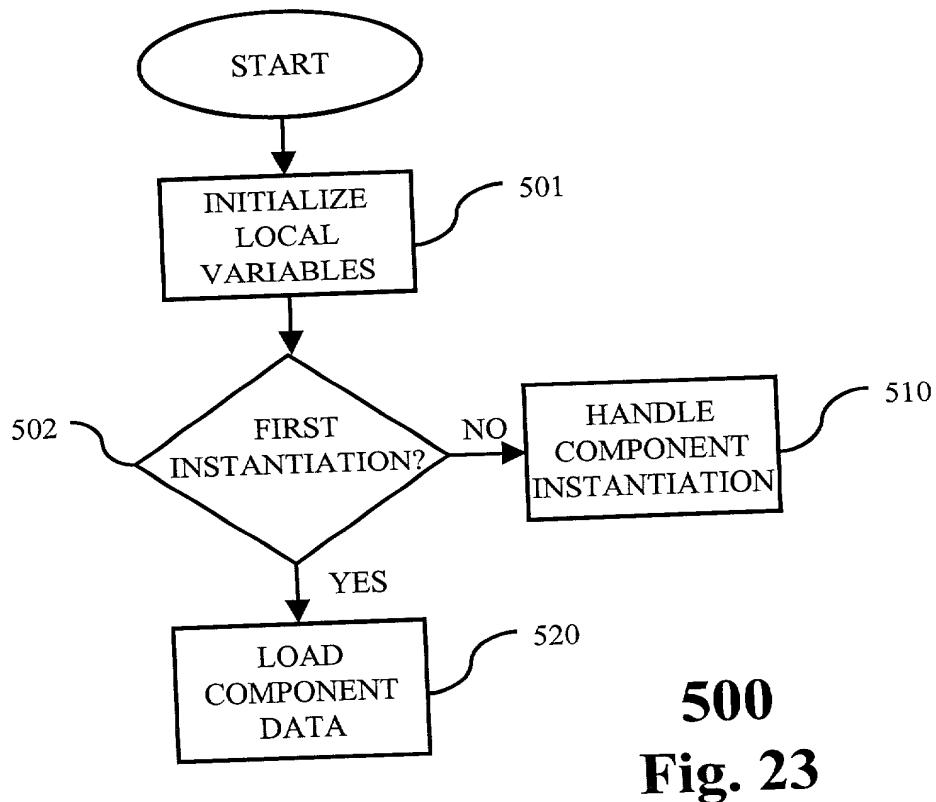
440
Fig. 20



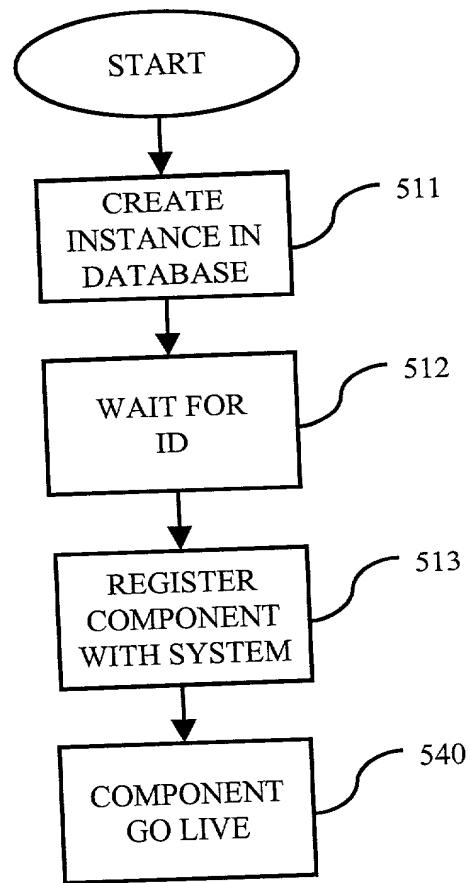
480
Fig. 21



490
Fig. 22

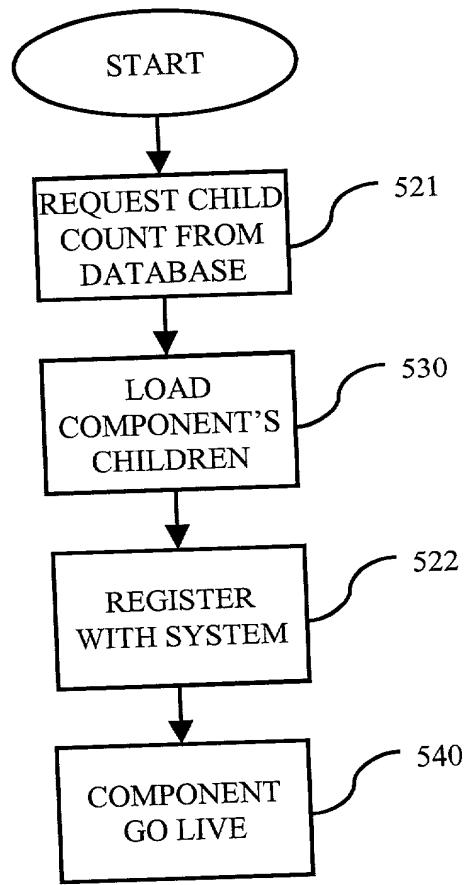


500
Fig. 23

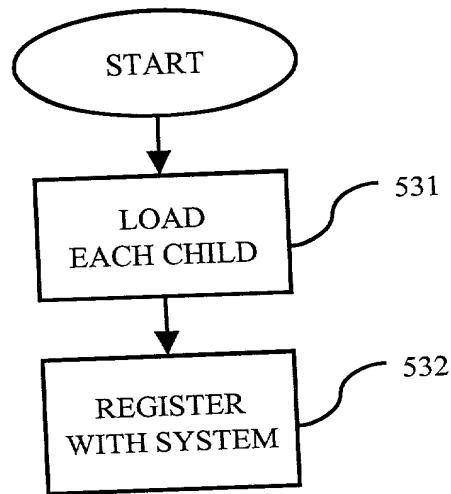


510

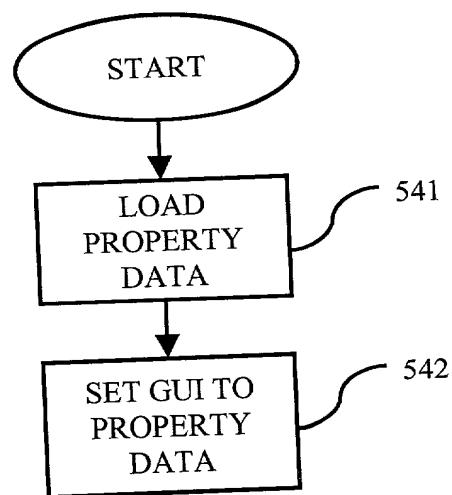
Fig. 24



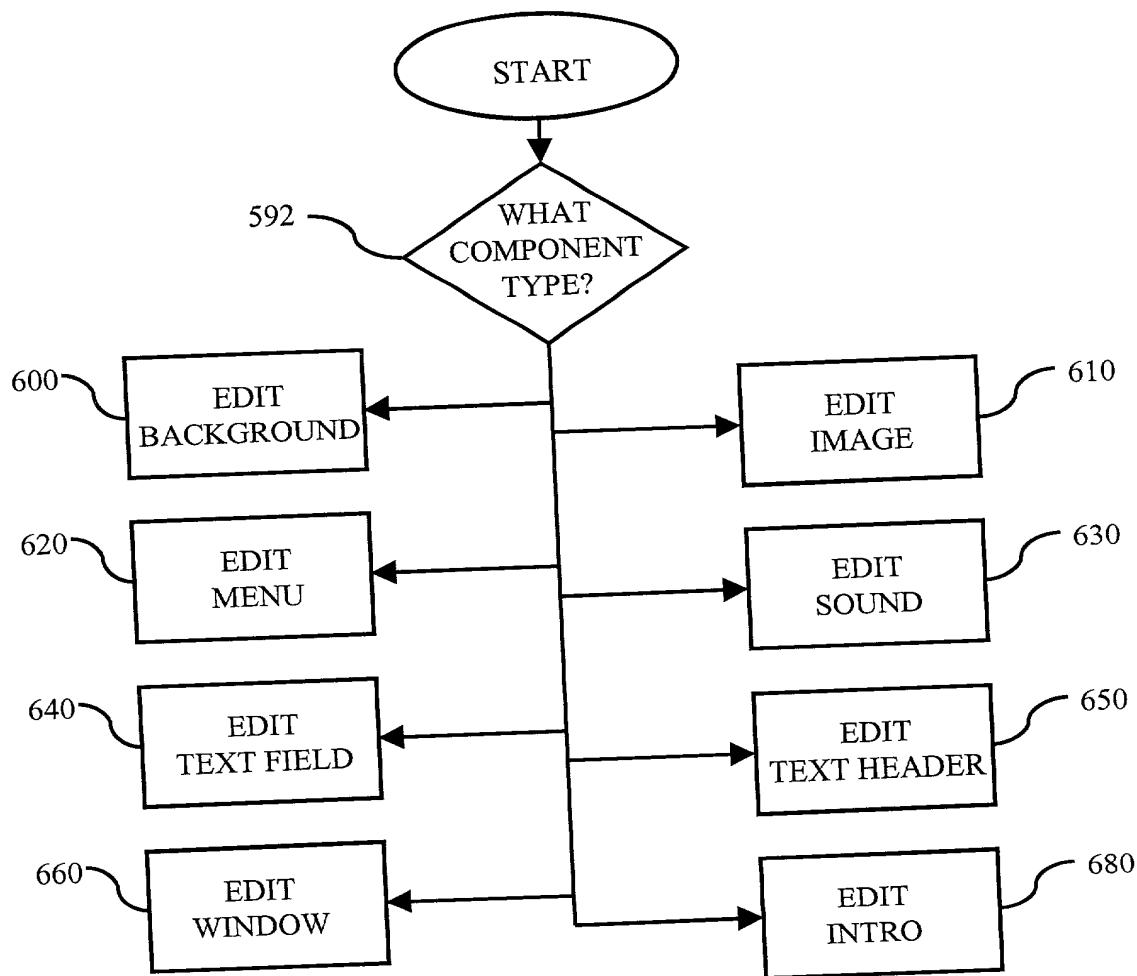
520
Fig. 25



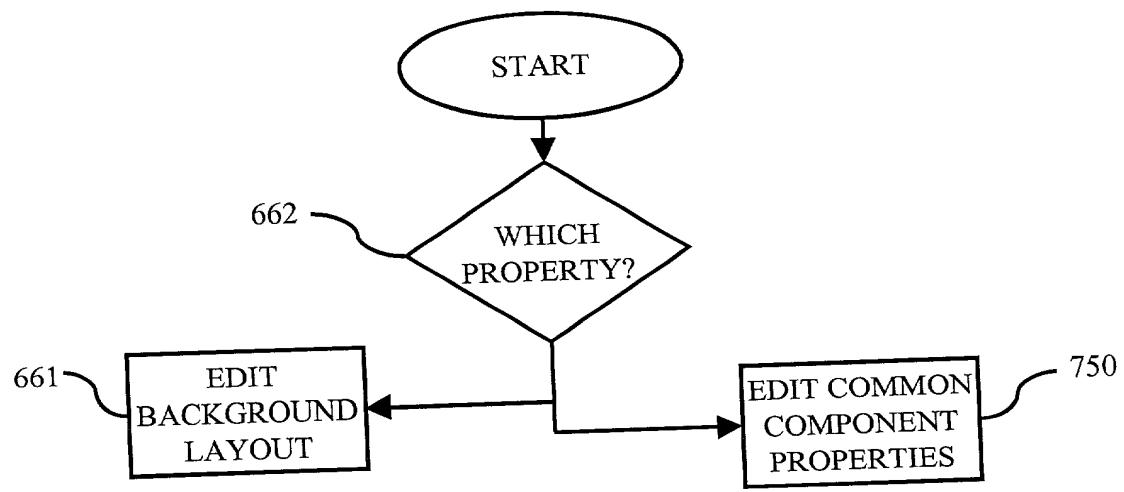
530
Fig. 26



540
Fig. 27

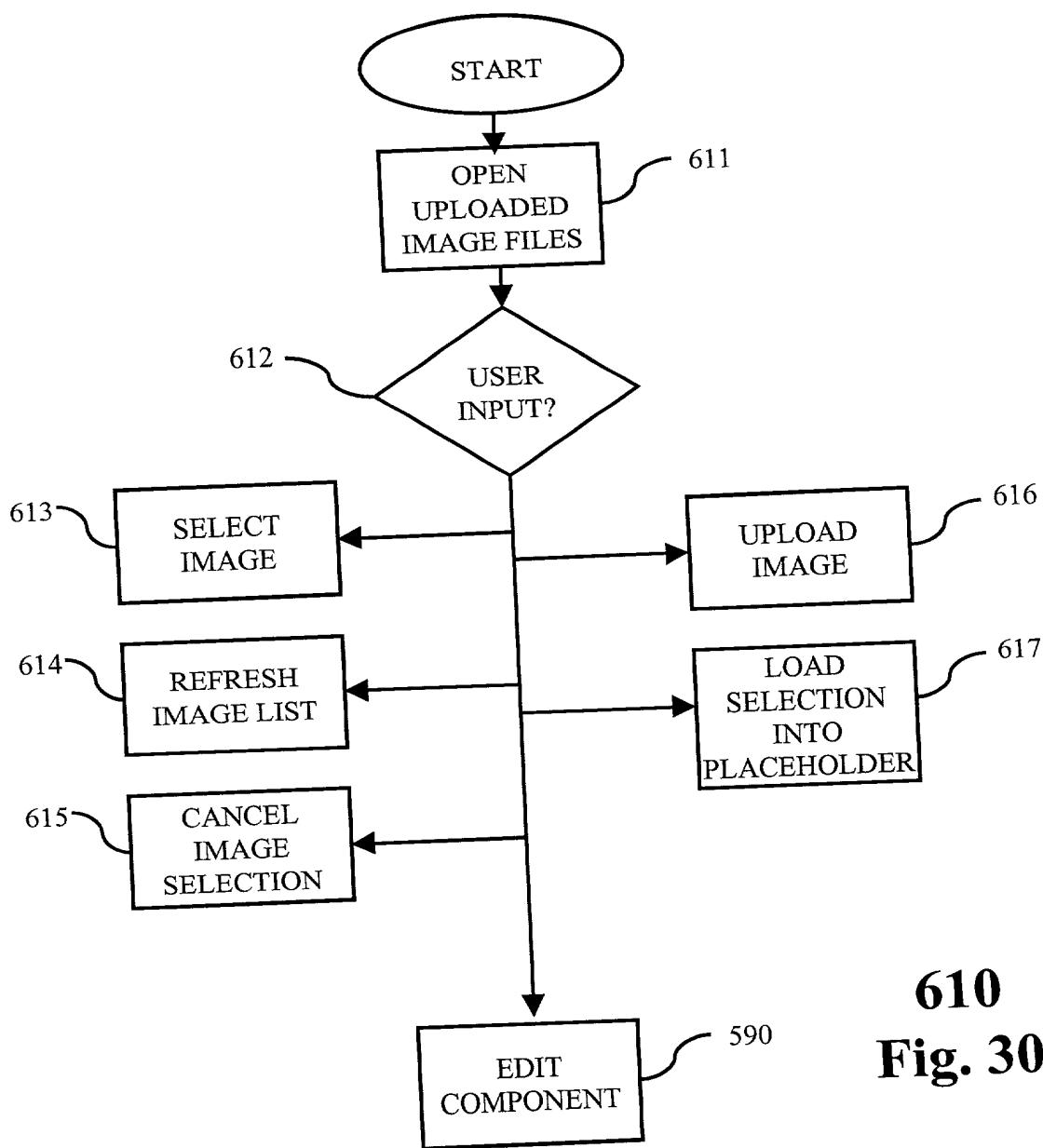


590
Fig. 28



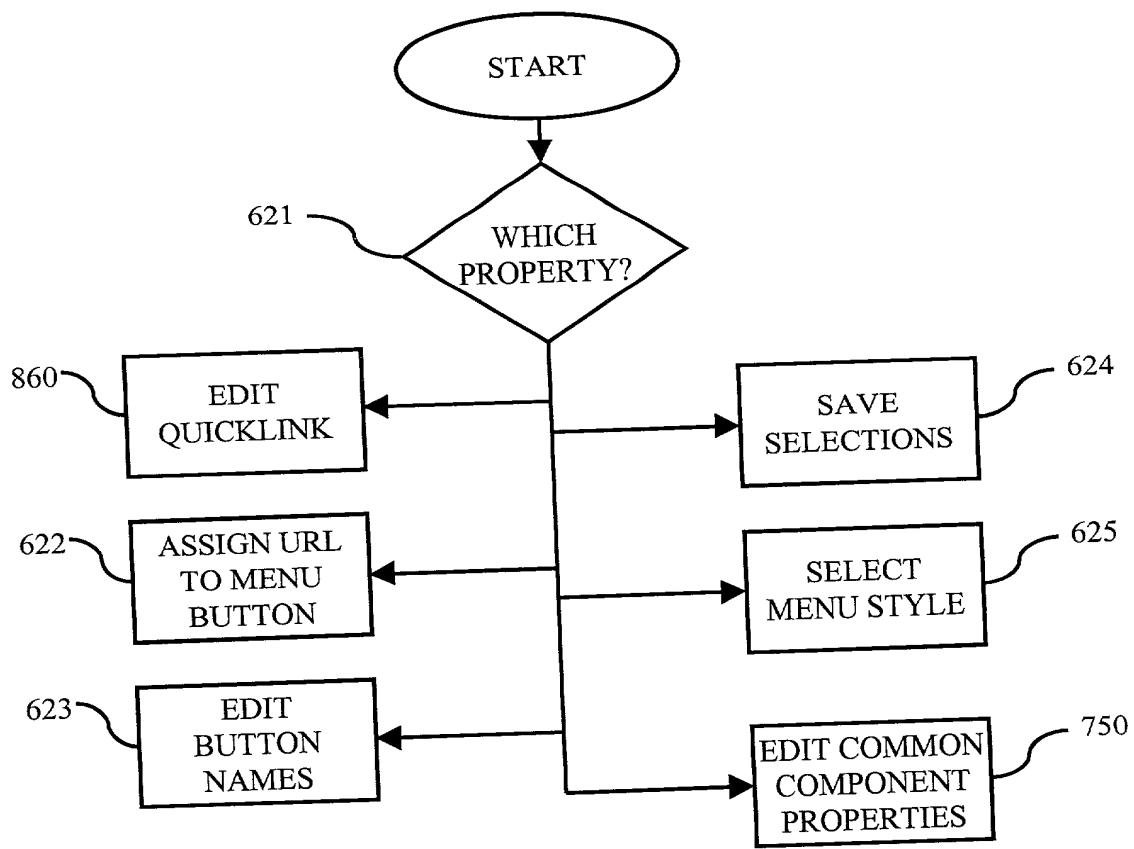
600

Fig. 29



610
Fig. 30

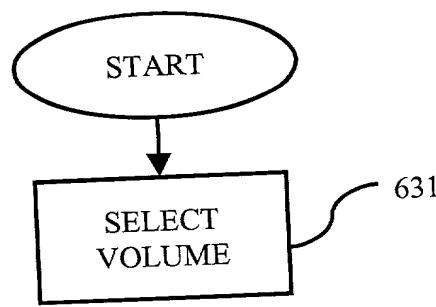
618 Asset Image Upload				
C1	Enter	✓		
C2	Has an intro been selected?	Yes	No	
C3	Has a loop been selected?	Yes	No	
C4	Which loop type?	A*	B*	C*
C5	Is loop count finished?	Yes	No	
C6	Has an outro been selected?			
A1	Set position, scale, rotation and alpha	✓		
A2	Play selected intro	✓	✓	✓
A3	Play selected loop	✓		
A4	Check loop count	✓		
A5	Decrement loop count	✓		
A6	Play outro	✓		
A7	Go to Rest frame			
	D I S P O S I T I O N	C2	C3	C4
		C5	C6	C7
		C8	C9	C10
		C11	C12	C13
		C14	C15	C16
		C17	C18	C19
		C20	C21	C22
		C23	C24	C25
		C26	C27	C28
		C29	C30	C31
		C32	C33	C34
		C35	C36	C37
		C38	C39	C40
		C41	C42	C43
		C44	C45	C46
		C47	C48	C49
		C50	C51	C52
		C53	C54	C55
		C56	C57	C58
		C59	C60	C61
		C62	C63	C64
		C65	C66	C67
		C68	C69	C70
		C71	C72	C73
		C74	C75	C76
		C77	C78	C79
		C80	C81	C82
		C83	C84	C85
		C86	C87	C88
		C89	C90	C91
		C92	C93	C94
		C95	C96	C97
		C98	C99	C100
		C101	C102	C103
		C104	C105	C106
		C107	C108	C109
		C110	C111	C112
		C113	C114	C115
		C116	C117	C118
		C119	C120	C121
		C122	C123	C124
		C125	C126	C127
		C128	C129	C130
		C131	C132	C133
		C134	C135	C136
		C137	C138	C139
		C140	C141	C142
		C143	C144	C145
		C146	C147	C148
		C149	C150	C151
		C152	C153	C154
		C155	C156	C157
		C158	C159	C160
		C161	C162	C163
		C164	C165	C166
		C167	C168	C169
		C170	C171	C172
		C173	C174	C175
		C176	C177	C178
		C179	C180	C181
		C182	C183	C184
		C185	C186	C187
		C188	C189	C190
		C191	C192	C193
		C194	C195	C196
		C197	C198	C199
		C200	C201	C202
		C203	C204	C205
		C206	C207	C208
		C209	C210	C211
		C212	C213	C214
		C215	C216	C217
		C218	C219	C220
		C221	C222	C223
		C224	C225	C226
		C227	C228	C229
		C230	C231	C232
		C233	C234	C235
		C236	C237	C238
		C239	C240	C241
		C242	C243	C244
		C245	C246	C247
		C248	C249	C250
		C251	C252	C253
		C254	C255	C256
		C257	C258	C259
		C260	C261	C262
		C263	C264	C265
		C266	C267	C268
		C269	C270	C271
		C272	C273	C274
		C275	C276	C277
		C278	C279	C280
		C281	C282	C283
		C284	C285	C286
		C287	C288	C289
		C290	C291	C292
		C293	C294	C295
		C296	C297	C298
		C299	C300	C301
		C302	C303	C304
		C305	C306	C307
		C308	C309	C310
		C311	C312	C313
		C314	C315	C316
		C317	C318	C319
		C320	C321	C322
		C323	C324	C325
		C326	C327	C328
		C329	C330	C331
		C332	C333	C334
		C335	C336	C337
		C338	C339	C340
		C341	C342	C343
		C344	C345	C346
		C347	C348	C349
		C350	C351	C352
		C353	C354	C355
		C356	C357	C358
		C359	C360	C361
		C362	C363	C364
		C365	C366	C367
		C368	C369	C370
		C371	C372	C373
		C374	C375	C376
		C377	C378	C379
		C380	C381	C382
		C383	C384	C385
		C386	C387	C388
		C389	C390	C391
		C392	C393	C394
		C395	C396	C397
		C398	C399	C400
		C401	C402	C403
		C404	C405	C406
		C407	C408	C409
		C410	C411	C412
		C413	C414	C415
		C416	C417	C418
		C419	C420	C421
		C422	C423	C424
		C425	C426	C427
		C428	C429	C430
		C431	C432	C433
		C434	C435	C436
		C437	C438	C439
		C440	C441	C442
		C443	C444	C445
		C446	C447	C448
		C449	C450	C451
		C452	C453	C454
		C455	C456	C457
		C458	C459	C460
		C461	C462	C463
		C464	C465	C466
		C467	C468	C469
		C470	C471	C472
		C473	C474	C475
		C476	C477	C478
		C479	C480	C481
		C482	C483	C484
		C485	C486	C487
		C488	C489	C490
		C491	C492	C493
		C494	C495	C496
		C497	C498	C499
		C500	C501	C502
		C503	C504	C505
		C506	C507	C508
		C509	C510	C511
		C512	C513	C514
		C515	C516	C517
		C518	C519	C520
		C521	C522	C523
		C524	C525	C526
		C527	C528	C529
		C530	C531	C532
		C533	C534	C535
		C536	C537	C538
		C539	C540	C541
		C542	C543	C544
		C545	C546	C547
		C548	C549	C550
		C551	C552	C553
		C554	C555	C556
		C557	C558	C559
		C560	C561	C562
		C563	C564	C565
		C566	C567	C568
		C569	C570	C571
		C572	C573	C574
		C575	C576	C577
		C578	C579	C580
		C581	C582	C583
		C584	C585	C586
		C587	C588	C589
		C590	C591	C592
		C593	C594	C595
		C596	C597	C598
		C599	C600	C601
		C602	C603	C604
		C605	C606	C607
		C608	C609	C610
		C611	C612	C613
		C614	C615	C616
		C617	C618	C619
		C620	C621	C622
		C623	C624	C625
		C626	C627	C628
		C629	C630	C631
		C632	C633	C634
		C635	C636	C637
		C638	C639	C640
		C641	C642	C643
		C644	C645	C646
		C647	C648	C649
		C650	C651	C652
		C653	C654	C655
		C656	C657	C658
		C659	C660	C661
		C662	C663	C664
		C665	C666	C667
		C668	C669	C670
		C671	C672	C673
		C674	C675	C676
		C677	C678	C679
		C680	C681	C682
		C683	C684	C685
		C686	C687	C688
		C689	C690	C691
		C692	C693	C694
		C695	C696	C697
		C698	C699	C700
		C701	C702	C703
		C704	C705	C706
		C707	C708	C709
		C710	C711	C712
		C713	C714	C715
		C716	C717	C718
		C719	C720	C721
		C722	C723	C724
		C725	C726	C727
		C728	C729	C730
		C731	C732	C733
		C734	C735	C736
		C737	C738	C739
		C740	C741	C742
		C743	C744	C745
		C746	C747	C748
		C749	C750	C751
		C752	C753	C754
		C755	C756	C757
		C758	C759	C760
		C761	C762	C763
		C764	C765	C766
		C767	C768	C769
		C770	C771	C772
		C773	C774	C775
		C776	C777	C778
		C779	C780	C781
		C782	C783	C784
		C785	C786	C787
		C788	C789	C790
		C791	C792	C793
		C794	C795	C796
		C797	C798	C799
		C800	C801	C802
		C803	C804	C805
		C806	C807	C808
		C809	C810	C811
		C812	C813	C814
		C815	C816	C817
		C818	C819	C820
		C821	C822	C823
		C824	C825	C826
		C827	C828	C829
		C830	C831	C832
		C833	C834	C835
		C836	C837	C838
		C839	C840	C841
		C842	C843	C844
		C845	C846	C847
		C848	C849	C850
		C851	C852	C853
		C854	C855	C856
		C857	C858	C859
		C860	C861	C862
		C863	C864	C865
		C866	C867	C868
		C869	C870	C871
		C872	C873	C874
		C875	C876	C877
		C878	C879	C880
		C881	C882	C883
		C884	C885	C886
		C887	C888	C889
		C890	C891	C892
		C893	C894	C895
		C896	C8	



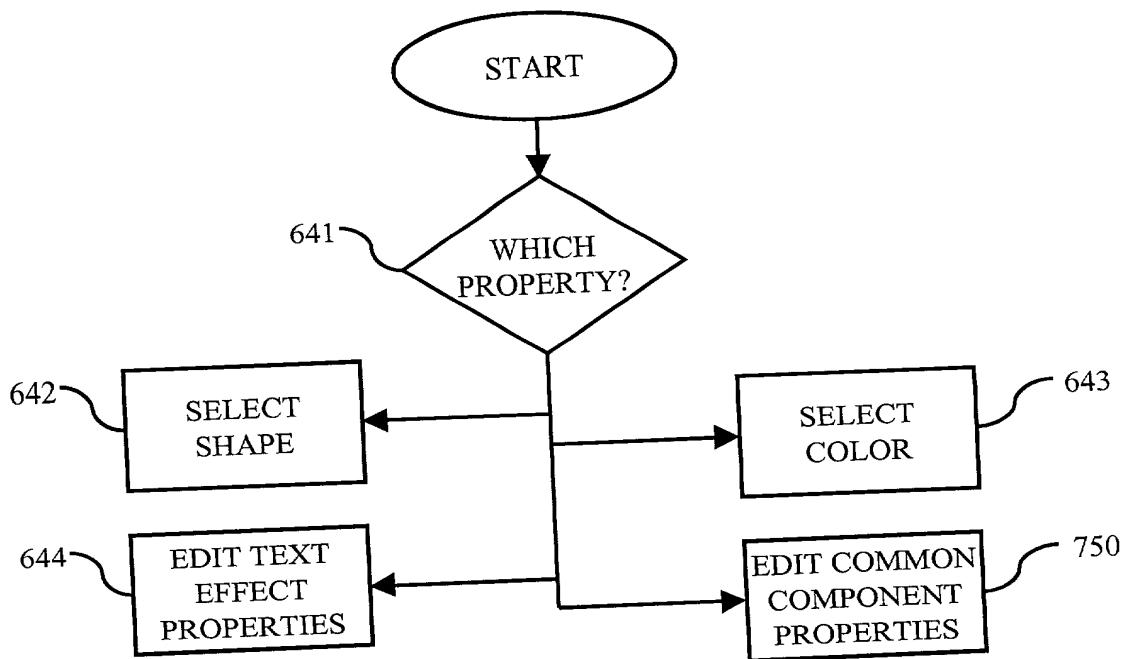
620
Fig. 32

628 Navigation Bar													
C1	Enter		X										
C2	Is number of buttons equal to number requested?			No	Yes								
A1	Set position, scale, rotation, and alpha		X										
A2	Receive button specific variables		X										
A3	Duplicate and position button		X										
A4	Select icon		X										
A5	Generate text field		X										
A6	Calculate size		X										
A7	Scale and position hit area		X										
DISPOSITION			C2	C2	Done								

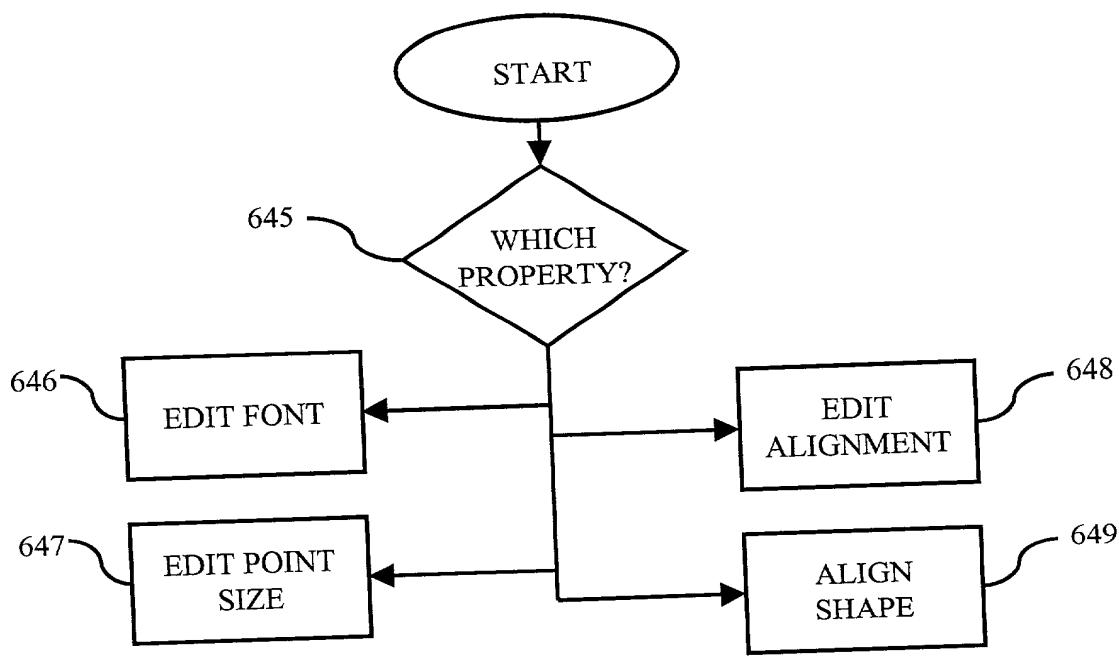
Fig. 33 – Navigation Bar Component 628



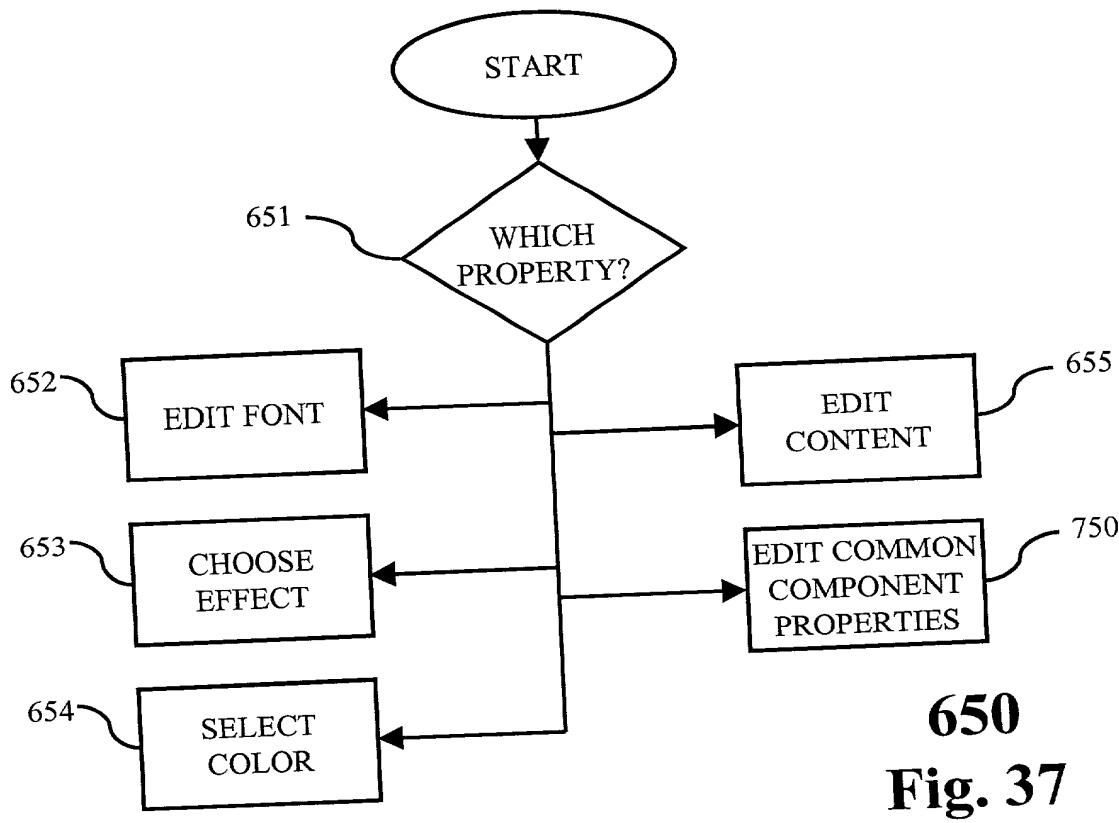
630
Fig. 34



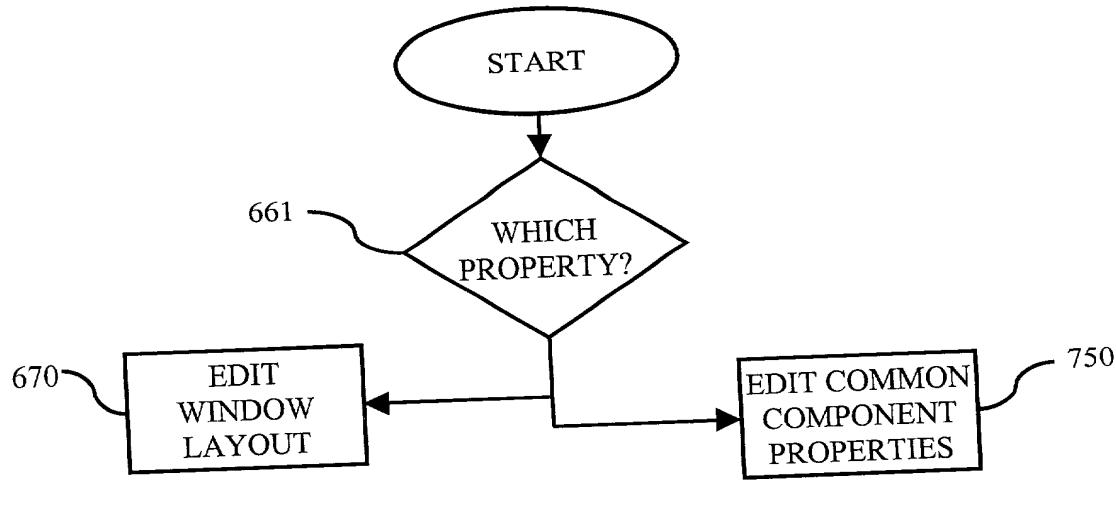
640
Fig. 35



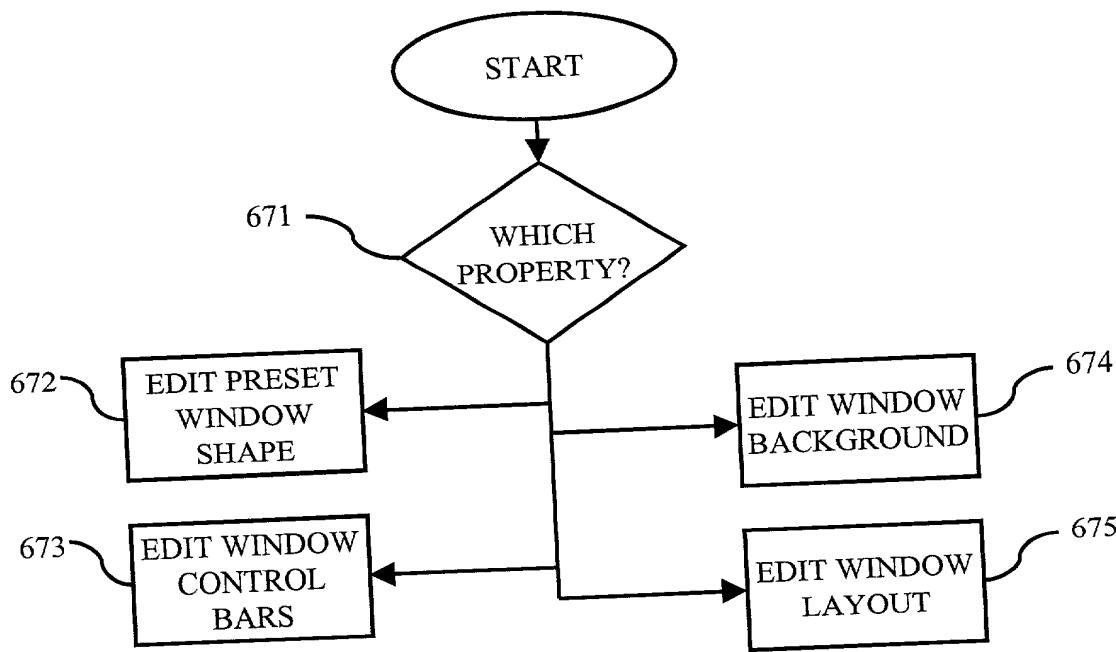
644
Fig. 36



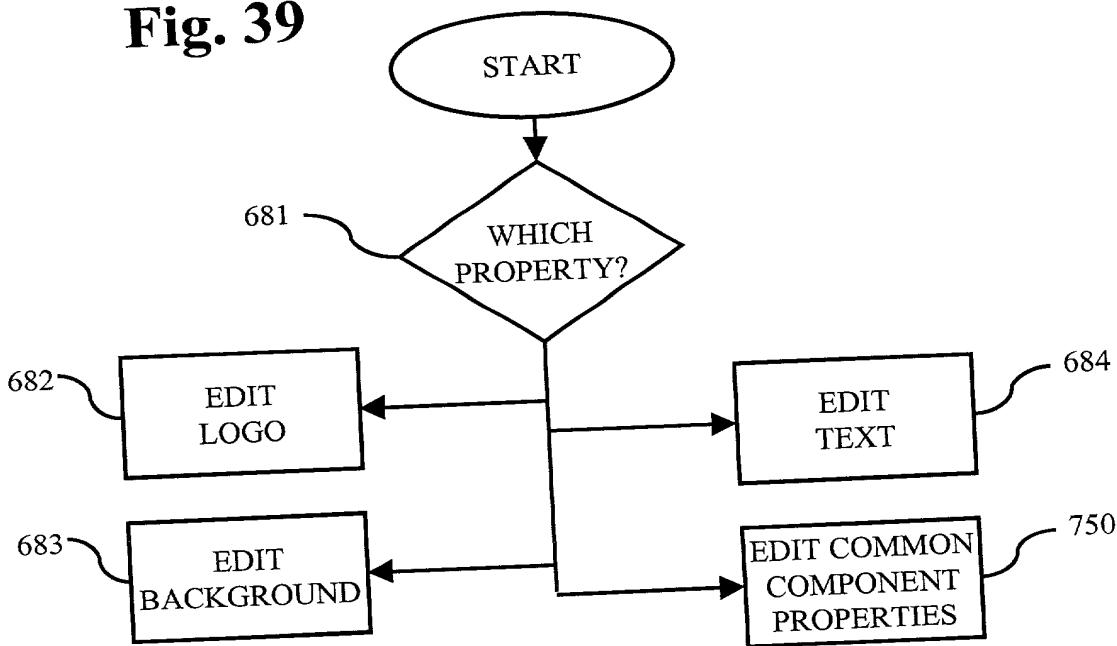
650
Fig. 37



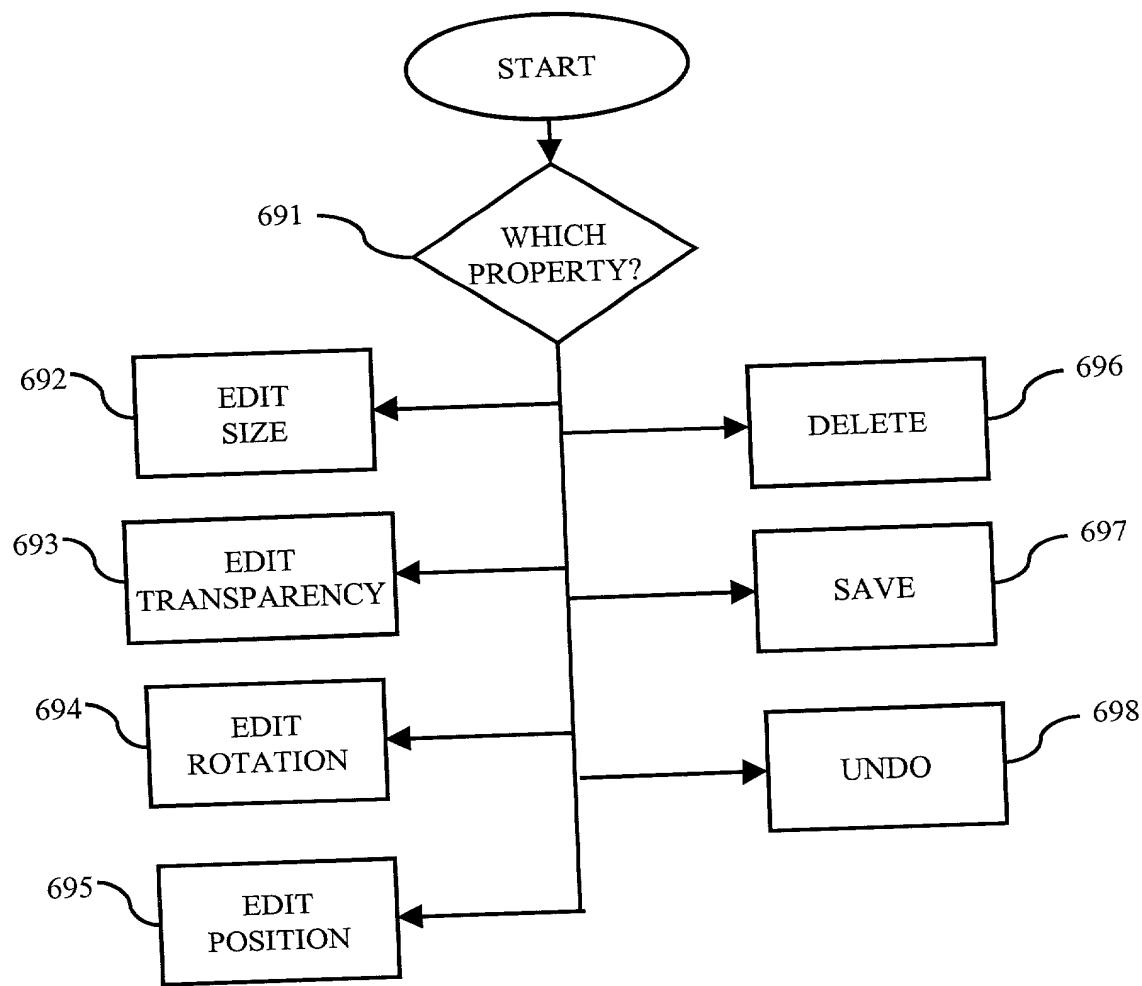
660
Fig. 38



670
Fig. 39

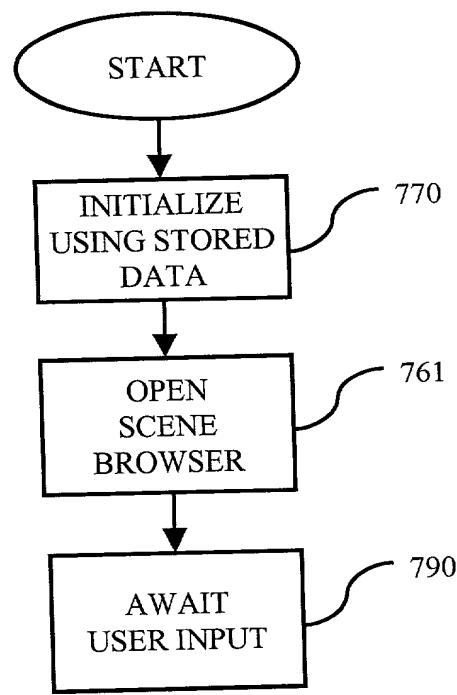


680
Fig. 40

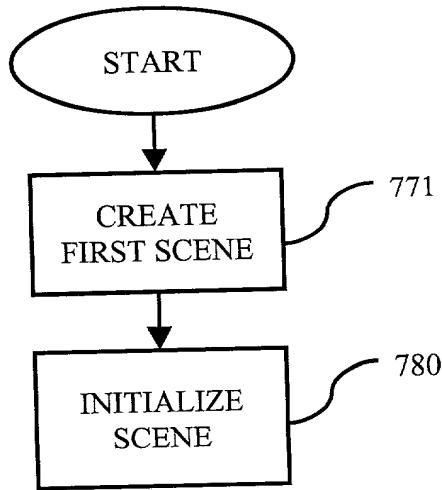


690

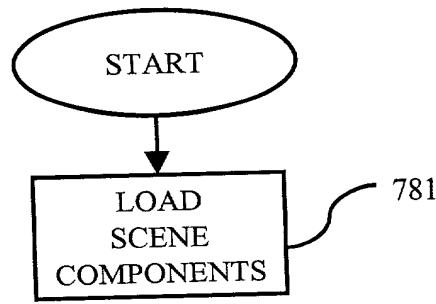
Fig. 41



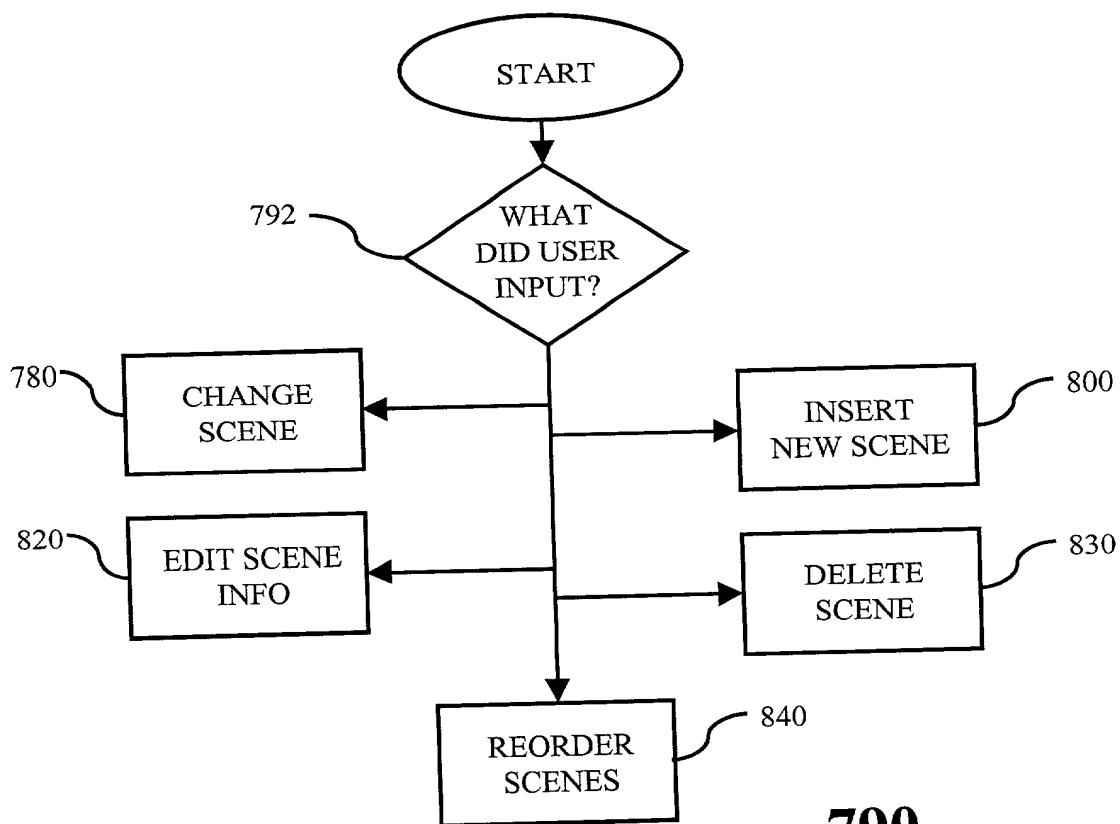
760
Fig. 42



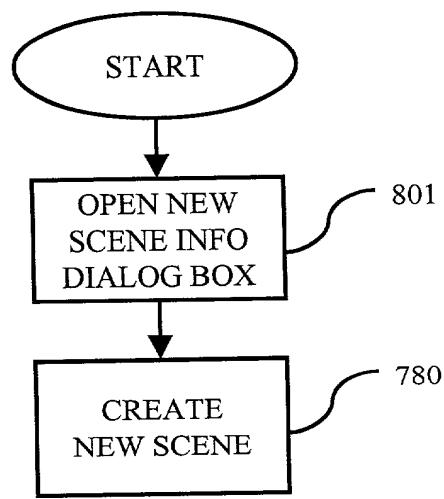
770
Fig. 43



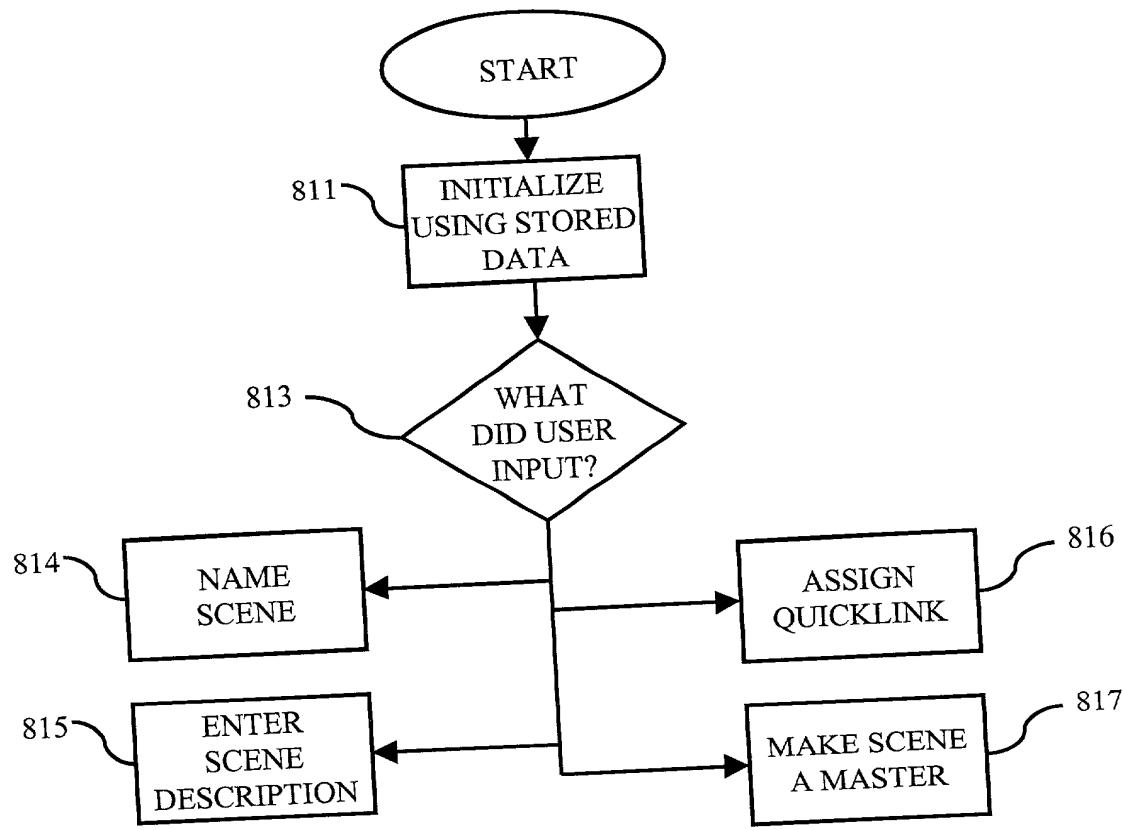
780
Fig. 44



790
Fig. 45

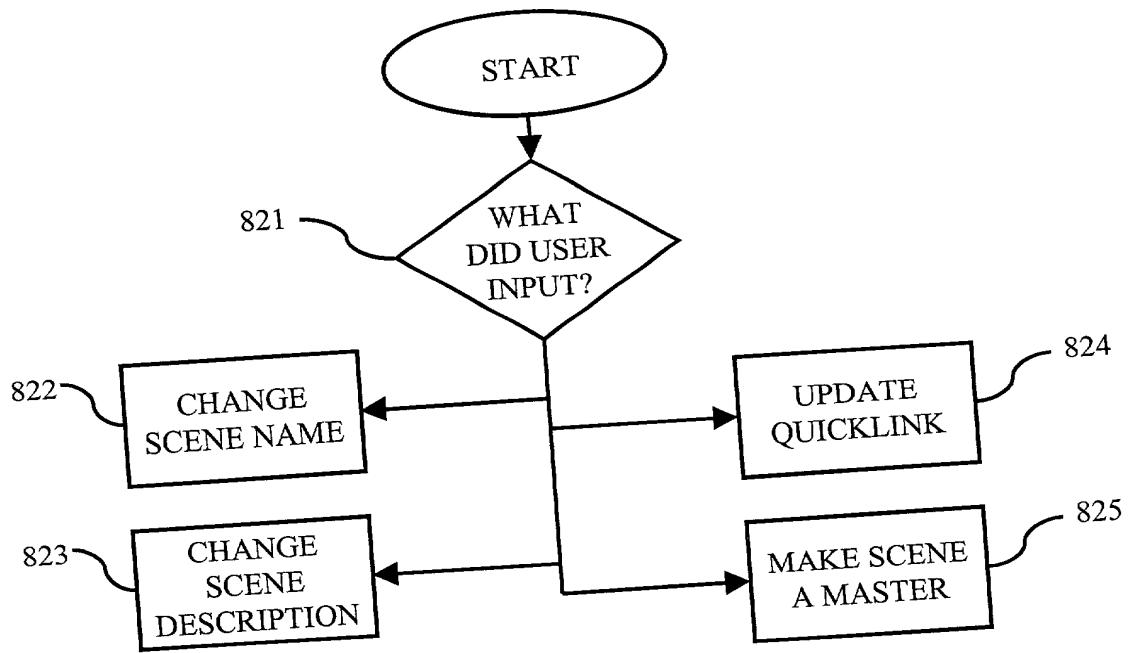


800
Fig. 46



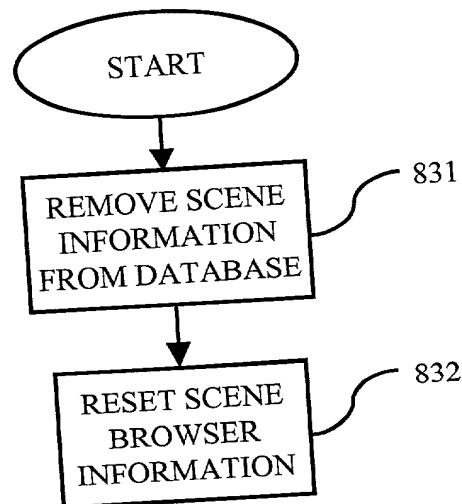
810

Fig. 47

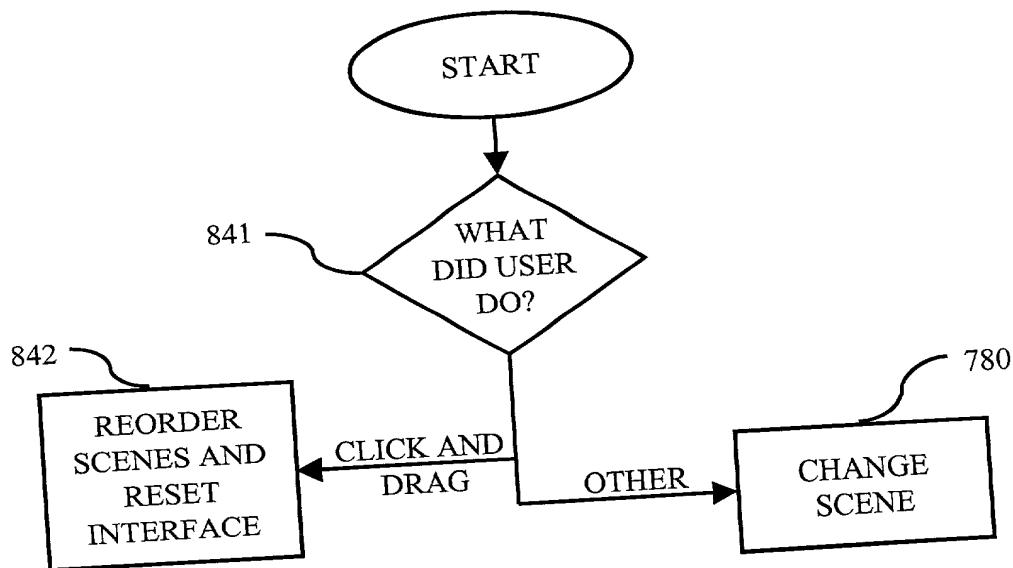


820

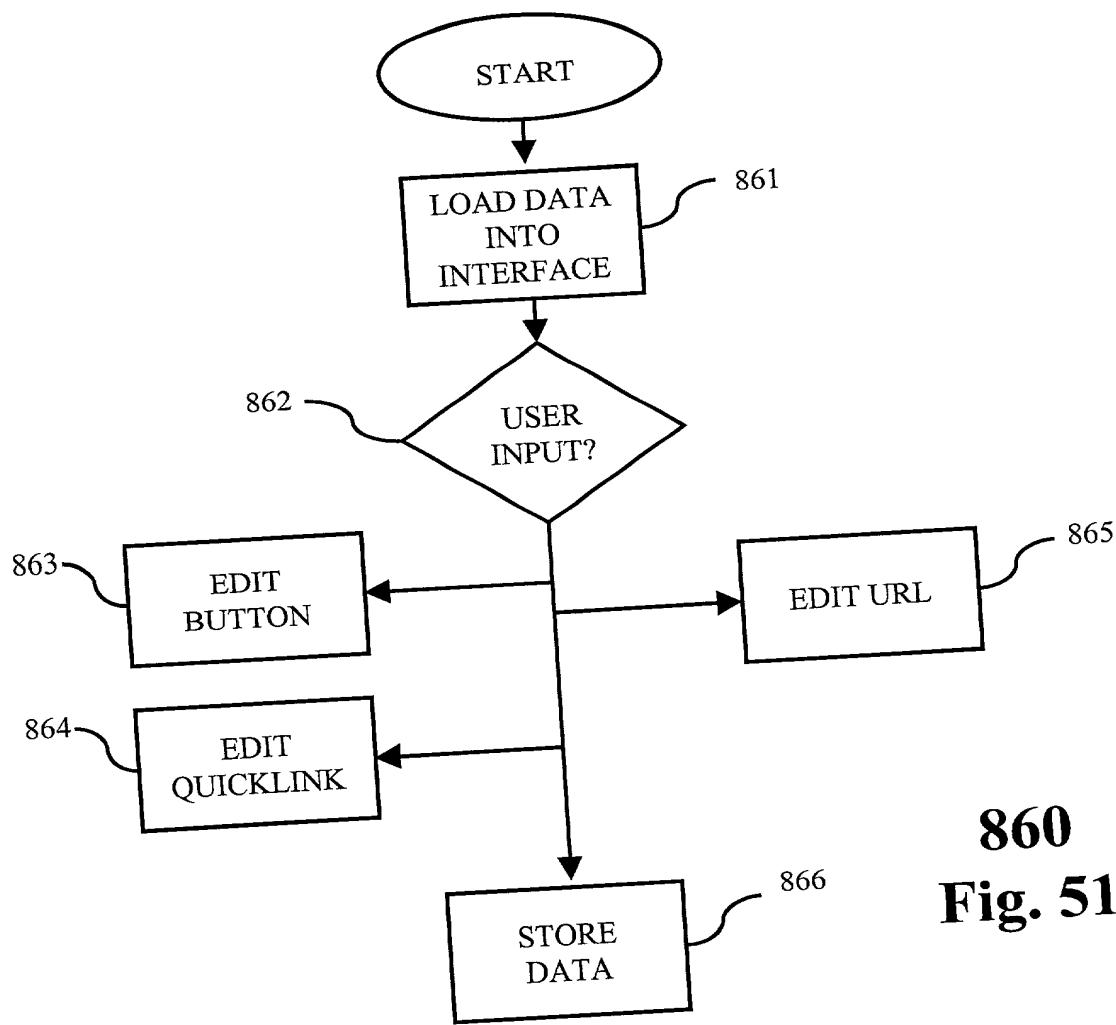
Fig. 48



830
Fig. 49



840
Fig. 50



860
Fig. 51

870	Component Browser														
C1	Is a Component Folder selected?		Yes	No											
C2	Does the Selected Folder contain other Folders?			Yes	No										
C3	Does the Selected Folder contain Components?				Yes	No									
C4	Is a Component selected?					Yes	No								
C5	Is the selected Component dropped over the Workspace?						Yes	No							
C6	Is a previous Header selected?								Yes	No					
A1	Add Header	X													
A2	Remove selected Header										X				
A3	Display selected Folder Name in Header	X													
A4	Hide previous List contents	X									X				
A5	Show selected List contents	X									X				
A6	Display Folder Icon with List entry(s)		X												
A6	Display Draggable Icon with List entry(s)			X											
A7	Display Component Dragger bullseye				X										
A8	Load Component into Project										X				
	Disposition	C2	C6	C3	C3	C4	C1	C5	C4	Done	C1	C1	C1		

Fig. 52 – Component Browser 870

871	Edit Size										
C1	Is a Preset Size selected?		Yes	No							
C2	Is Size selected with the Slider?			Yes	No						
C3	Is Size value entered in the textfield?				Yes	No					
C4	Is the Return key pressed?						Yes	No			
A1	Set user-defined Size		X	X			X				
A2	Set default Size			X	X		X				
	Disposition		Done	C2	Done	C3	C4	Done	Done	C1	

Fig. 53 – Edit Size 871

872	Edit Transparency								
C1	Is a Preset Transparency selected?		Yes	No					
C2	Is Transparency selected with the Slider?			Yes	No				
C3	Is Transparency value entered in the textfield?				Yes	No			
C4	Is the Return key pressed?						Yes	No	
A1	Set user-defined Transparency		X	X			X		
A2	Set default Transparency			X	X	X			
	Disposition		Done	C2	Done	C3	C4	Done	C1

Fig. 54 – Edit Transparency 872

873	Edit Rotation													
C1	Is a Preset Rotation selected?		Yes	No										
C2	Is Rotation selected with the Slider?			Yes	No									
C3	Is Rotation value entered in the textfield?				Yes	No								
C4	Is the Return key pressed?						Yes	No						
A1	Set user-defined Rotation		X		X			X						
A2	Set default Rotation			X		X								
	Disposition		Done	C2	Done	C3	C4	Done	Done	Done	C1			

Fig. 55 – Edit Rotation 873

874	Edit Position													
C1	Is default Position button pressed?		Yes	No										
C2	Is Position changed with the draggable icon?			Yes	No									
C3	Is Position changed with the arrow keys?				Yes	No								
C4	Are Position values entered in the textfields?						Yes	No						
C5	Is Return/Enter pressed?								Yes	No				
A1	Set user-defined Position				X		X			X				
A2	Set default Position		X			X		X		X				
	Disposition		Done	C2	Done	C3	Done	C4	C5	Done	Done	C1		

Fig. 56 – Edit Position 874

875	Edit Color													
C1	Is a Color swatch selected?		Yes	No										
C2	Is Brightness selected with the Slider?				Yes	No								
C3	Is Brightness value entered in the textfield?						Yes	No						
C4	Is the Return key pressed?								Yes	No				
A1	Set user-defined Color		X											
A2	Set default Color			X										
A3	Set user-defined Brightness				X					X				
A3	Set default Brightness					X			X					
	Disposition		C2	C2	C3	C3	C4	Done	Done	C1				

Fig. 57 – Edit Color 875

876	Edit Selection													
C1	Is Selection selected?		Yes	No										
C2	Is Next/Previous arrow pressed?				Next	Prev	None							
A1	Set user-defined Selection		X											
A2	Set default Selection						X							
A3	Show next Selection				X									
A4	Show previous Selection					X								
	Disposition		Done	C2	C1	C1	Done							

Fig. 58 – Edit Selection 876

877	Edit Content (for Paragraph component)														
C1	Is Textfield content set?		Yes	No											
C2	Is Font selected?			Yes	No										
C3	Is Size selected?				Yes	No									
C4	Is Align selected?						Yes	No							
C5	Is Shape selected?								Yes	No					
C6	Is Apply pressed?										Yes	No			
A1	Set user-defined Textfield content	X													
A2	Set default Textfield content		X												
A3	Set selected Font			X											
A4	Set default Font				X										
A5	Set selected Size					X									
A6	Set default Size						X								
A7	Set selected Alignment							X							
A8	Set default Alignment								X						
A9	Set selected Shape									X					
A11	Set attributes										X				
	Disposition		C2	C2	C3	C3	C4	C4	C5	C5	C6	C6	Done	C1	

Fig. 59 – Edit Content (for Paragraph components) 877

Fig. 60 – Edit QuickLink (for Button components) 878

Fig. 61 - Edit Selection (for Button components) 879

880	Edit Content (for Line Effects components)													
C1	Is Textfield content set?		Yes	No										
C2	Is Font selected?			Yes	No									
C3	Is Apply pressed?					Yes	No							
A1	Set user-defined Textfield content		X											
A2	Set default textfield content			X										
A3	Set selected Font				X									
A4	Set default Font					X								
A5	Set attributes						X							
	Disposition		C2	C2	C3	C3	Done	C1						

Fig. 62 – Edit Content (for Line Effects components) 880

881	Edit Soundtrack													
C1	Is Soundtrack selected?		Yes	No										
C2	Is Volume level set?			Yes	No									
C3	Volume On/Off?					On	Off							
A1	Set default Soundtrack		X											
A2	Set Selected Soundtrack	X												
A3	Set default Volume level				X									
A4	Set selected Volume level			X										
A5	Play Soundtrack					X								
A6	Stop All Sounds						X							
	Disposition	C2	C2	C3	C3	Done	Done							

Fig. 63 – Edit Soundtrack 881

882	Edit User Assets										
C1	Is a User Asset selected?	Yes	No								
C2	Is Asset Data selected?			Yes			No				
C3	Is Remove selected?				Yes			No			
C4	Is Upload selected?					Yes		No			
C5	Is Refresh selected?						Yes		No		
C6	Is Accept selected							Yes	No		
A1	Display icon of selected User Asset	X									
A2	Display icon of first User Asset		X								
A3	Show/Hide Asset Data			X							
A4	Remove selected User Asset				X						
A5	Launch Upload Assets pop-up window					X					
A6	Reload User Asset information						X				
A7	Set selected User Asset							X			
	Disposition	C2	C2	C3	Done	C1	C1	Done	C1		

Fig. 64 – Edit User Assets 882

883	Edit Content (for Character Effects component)										
C1	Is Textfield Content set?		Yes	No							
C2	Is Font Selected?				Yes	No					
C3	Is Apply button pressed?						Yes	No			
A1	Set Textfield content	X									
A2	Set default Textfield content		X								
A3	Set selected Font			X							
A4	Set default Font				X						
A5	Apply settings					X					
	Disposition	C2	C2	C3	C3	Done	C1				

Fig. 65 – Edit Content (for Character Effects components) 883

Fig. 66 – Edit Content (for Movie components) 884

Fig. 67 – Edit Content (for Window components) 885

886	Edit Content (for Header component)									
C1	Is Textfield content set?		Yes	No						
C2	Is Font selected?			Yes	No					
C3	Is Size selected?					Yes	No			
C4	Is Apply pressed?							Yes	No	
A1	Set user-defined textfield content		X							
A2	Set default textfield content			X						
A3	Set selected Font				X					
A4	Set default Font					X				
A5	Set selected Size						X			
A6	Set default Size							X		
A7	Set attributes								X	
	Disposition		C2	C2	C3	C3	C4	C4	Done	C1

Fig. 68 – Edit Content (for Header components) 886

887 Scale/Position Handles															
C1	Is a component selected in the Layers Browser?		Yes	No											
C2	Is the hit area selected?			Yes	No										
C3	Is a corner handle selected?				Yes	No									
C4	Is the left or right handle selected?						Yes	No							
C5	Is the top or bottom handle selected?								Yes	No					
C6	Is the component QuickLinkable?										Yes	No			
A1	Set user-defined component	X													
A2	Set default component (Layer 1 component)		X												
A3	Reposition component			X											
A4	Resize X-scale of component				X		X								
A5	Resize Y-scale of component					X				X					
A6	Show QuickLink information											X			
A7	Hide QuickLink information													X	
	Disposition		C2	C2	C3	C3	C6	C4	C6	C5	C6	C6	Done	Done	

Fig. 69 – Scale/Position Handles 887

888 Start Depth Browser															
C1	Is Depth Browser open?				No	Yes									
C2	Open tab clicked?		No	Yes											
A1	Open Depth Browser					✓									
A2	Load component list from current scene					✓									
A3	Display components in top to bottom (front to back) order					✓	✓								
A4	Wait for user input 889					✓	✓								
	D I S P O S I T I O N				C1	C1	Done								

Fig. 70 – Start Depth Browser 888

889 Wait User Input Depth Browser		Drag In New Comp	Drag Comp	Vis Btn	Lock Btn	Sel Comp	Cls Tab
C1	User Input?						
A1	Place new component on top	✓					
A2	Change depth of component being dragged (cannot drag a locked component)		✓				
A3	Toggle component's visibility			✓			
A4	Toggle component lock				✓		
A5	Apply Unified GUI interfaces to selected component (871 to 887 as applicable)					✓	
A6	Reload component list from current scene	✓	✓				
A7	Redisplay components in top to bottom (front to back) order	✓	✓	✓			
A8	Close Depth Browser Panel						✓
D I S P O S I T I O N		C1	C1	C1	C1	C1	Done

Fig. 71 – Wait User Input 889

890 Layers Window Opening and Closing									
C1	Is window being opened?	Y	N						
C2	Is (new) scene selected?			Y	N				
C3	Is window being closed?					Y	N		
C4	Is mouse over the window?							Y	N
A1	Open Window	✓							
A2	Load objects from scene		✓						
A3	Close Window				✓				
A4	Select Object (891)						✓		
D I S P O S I T I O N		C2	C1	C3	C3	C1	C4	C2	C2

Fig. 72 – Layers Window Opening and Closing 890

891 Layers Window Select Object									
C1	Is mouse over an unselected, unlocked object?	Y	N						
C2	Was mouse clicked?		Y	N					
C3	Is mouse over a selected, unlocked object?				Y	N			
C4	Was mouse clicked?						Y	N	
A1	Highlight object line in orange	✓							
A2	Unselect previously selected object, if any Unhighlight object back to gray Remove arrows from life bar			✓					
A3	Select new object, Highlight object line in green, Place arrows on life bar			✓					
A4	Edit Object (893)			✓				✓	
A5	Select Scene Time (892)							✓	✓
D I S P O S I T I O N		C2	C3	D	C3	C4	D	D	D

Fig. 73 – Select Object 891

892 Layers Window Select Scene Time									
C1	Is mouse over total scene time?	Y	N						
C2	Was mouse clicked?			Y	N				
C3	Was valid time entered?					Y	N		
A1	Pop up Scene Time entry window	✓							
A2	Accept user typein of scene time			✓			✓		
A3	Process new scene time Repaint tickmarks in window header Repaint object life bars					✓			
A4	Remove Scene Time entry window				✓	✓			
D I S P O S I T I O N		C2	D	C3	D	D	C3		

Fig. 74 – Select Scene Time 892

893 Layers Window Edit Object									
C1	Was mouse clicked?	Y	N						
C2	Is drag being performed?			Y	N				
C3	Was visibility button clicked?					Y	N		
C4	Was lock button clicked?							Y	N
A1	Process Drag (894)			✓					
A2	Process Vis Button (895)					✓			
A3	Process Lock Button (896)						✓		
D I S P O S I T I O N		C2	D	D	C3	D	C4	D	D

Fig. 75 – Edit Object 893

894 Layers Window Process Drag		Y	N								
C1	Is object being dragged vertically?		Y	N							
C2	Is object life bar left arrow being dragged?			Y							
C3	Is object life bar right arrow being dragged?				Y						
C4	Is object life bar as a whole being dragged?						Y	N			
A1	Move object to new front to back display position	✓									
A2	Redisplay scene in correct order	✓									
A3	Change object start time and duration Resize life bar accordingly			✓							
A4	Change object end time and duration Resize life bar accordingly					✓					
A5	Move object start and end times in tandem Duration remains unchanged Reposition life bar accordingly						✓				
D I S P O S I T I O N		D	D	D	D	D	D				

Fig. 76 – Process Drag 894

895 Layers Window Process Vis Button		Y	N								
C1	Was object visible?		Y	N							
A1	Make object invisible	✓									
A2	Gray out button	✓									
A3	Make object visible		✓								
A4	Turn button green		✓								
D I S P O S I T I O N		D	D								

Fig. 77 – Process Visibility Button 895

896 Layers Window Process Lock Button		Y	N							
C1	Was object locked?		Y							
A1	Unlock object	✓								
A2	Set gray dot for button	✓								
A3	Lock object		✓							
A4	Set gray lock icon for button		✓							
	D I S P O S I T I O N	D	D							

Fig. 78 – Process Lock Button 896

900	Asset Manager					
C1	Enter	Yes				
A1	Load Asset Manager Components (1100)	✓				
A2	Init all Asset Manager Components	✓				
A3	Start Idle Loop (901)	✓				
	D I S P O S I T I O N		Done			

Fig. 79 – Asset Manager 900

901	Idle Loop					
C1	Enter at frame 1	Yes				
C2	Does frame number equal control variable?		No	Yes		
C3	Is this the last frame?		No	Yes		
A1	Advance to next frame	✓				
A2	Call Load Check (920)	✓	✓			
A3	Call Request Scanner (910)	✓				
A4	Jump back to frame 1			✓	✓	
	D I S P O S I T I O N		C2	C2	C1	C1

Fig. 80 – Idle Loop 901

910	Request Scanner								
C1	Enter	Yes							
C2	Are there requests to process?		No		Yes				
L3	Loop through all scheduled requests			Ent	Loop	Exit			
C4	More to process?			Yes	No				
A1	Call Scheduler (930)	✓							
A2	Pick Request			✓					
A3	Process Request (911)			✓					
D I S P O S I T I O N				C2	Done	L3 Loop	L3 Loop	L3 Exit	Done

Fig. 81 – Request Scanner 910

911	Request Scanner - continued								
	Process Request								
C1	General request type?		Reg	Load	Play	Posn	State	Loc	Glob
								Vol	Vol
A1	Call Registration Request Processor (940)	✓							
A2	Call Load Request Processor (950)		✓						
A3	Call Play Request Processor (960)			✓					
A4	Call Position Request Processor (970)				✓				
A5	Call State Request Processor (980)					✓			
A6	Call Local Volume Request Processor (990)						✓		
A7	Call Global Volume Request Processor (1000)							✓	
A8	Return error response to requestor								✓
D I S P O S I T I O N				Done	Done	Done	Done	Done	Done

Fig. 82 – Process Request 911

920 Load Check		Ent	Loop		Exit	
L1	Loop through all clips being loaded					
C2	More clips to check?		Yes	No		
C3	Are all frames loaded?				Yes	No
A1	Pick clip		✓			
A2	Get frames loaded for clip		✓			
A3	Set slot identification into clip			✓		
A4	Mark clip load finished in check list			✓		
A5	Set clip state to loaded			✓		
A6	Remove finished clips from check list				✓	
D I S P O S I T I O N		L1 Loop	C3	L1 Exit	L1 Loop	L1 Done

Fig. 83 – Load Check 920

930 Scheduler		Yes					
C1	Enter						
C2	Are intensive task(s) in progress?		Yes	No			
A1	Transfer requests from reschedule list to schedule list	✓					
A2	Set reschedule list empty	✓					
A3	Schedule Intensive Task Request (931)		✓				
A4	Process Request List (932)			✓			
A5	Empty request list			✓			
D I S P O S I T I O N		C2	Done	Done			

Fig. 84 – Scheduler 930

931	Scheduler Schedule Intensive Task Request							
C1	Is there an internal request for an intensive task?	Yes	No					
C2	Is there a request from an intensive task?		Yes	No				
A1	Add Intensive Task request to schedule list	✓	✓					
	D I S P O S I T I O N	Done	Done	Done				

Fig. 85 – Schedule Intensive Task 931

932	Scheduler Process Request List							
L1	Loop through request list	Ent		Loop		Exit		
C2	More slots in request list?		Yes	No				
C3	Is there an internal request in the slot?				Yes	No		
C4	Is there an external request in the slot?					Yes	No	
A1	Pick slot	✓						
A2	Add request to schedule list			✓	✓			
	D I S P O S I T I O N	L1 Loop	C3 Exit	L1 Loop	L1 Loop	L1 Loop	L1 Loop	Done

Fig. 86 – Process Request List 932

940 Registration Request Processor							
C1	Request type?	Reg	Re reg	Un reg	Query Reg	No Req	Oth
A1	Process Register (941)	✓					
A2	Process Reregister (942)		✓				
A3	Process Unregister (943)			✓			
A4	Process Query Register (944)				✓		
A5	Ignore No Request					✓	
A6	Return error response to requestor						✓
D I S P O S I T I O N		Done	Done	Done	Done	Done	Done

Fig. 87 – Registration Request Processor 940

941 Registration Request Processor Process Register							
C1	Enter	Yes					
C2	Any argument errors?		No	Yes			
A1	Assign slot and initialize it	✓					
A2	Get arguments	✓					
A3	Assign arguments to the slot including: name, type, URL, instance name, attributes, permissions, parent, obey, URLs for cycle and bandwidth		✓				
A4	Set status as registered		✓				
A5	Return OK response to requestor		✓				
A6	Return error response to requestor			✓			
D I S P O S I T I O N		C2	Done	Done			

Fig. 88 – Process Register 941

942 Registration Request Processor Process Reregister										
C1	Is clip currently registered?	Yes		No						
C2	Do permissions allow reregistration?	No	Yes							
C3	Any argument errors?			No		Yes				
A1	Assign slot and initialize it			✓						
A2	Get arguments	✓		✓						
A3	Assign arguments to the slot including: name, type, URL, instance name, attributes, permissions, parent,			✓						
A4	Set status as registered			✓						
A5	Return OK response to requestor			✓						
A6	Return error response to requestor	✓			✓					
D I S P O S I T I O N		Done	C3	C3	Done	Done				

Fig. 89 – Process Reregister 942

943 Registration Request Processor Process Unregister										
C1	Is clip currently registered?	Yes		No						
C2	Do permissions allow unregistration?	No	Yes							
C3	Is clip playing?	No		Yes						
C4	Is clip loaded?	No	Yes							
A1	Stop clip play			✓						
A2	Unload clip	✓		✓						
A3	Set status as unregistered	✓	✓	✓						
A4	Return OK response to requestor	✓	✓	✓						
A5	Return error response to requestor	✓			✓					
D I S P O S I T I O N		Done	Done	Done	Done	Done				

Fig. 90 – Process Unregister 943

944 Registration Request Processor Process Query Register							
C1	Query by slot?	Yes	No				
C2	Query by name?	Yes	No				
C3	Query by instance name?		Yes	No			
C4	Query by URL?			Yes	No		
C5	Was clip found?					Yes	No
C6	Is clip registered?					Yes	No
A1	Find slot	✓					
A2	Find clip by name	✓					
A3	Find clip by instance name		✓				
A4	Find clip by URL			✓			
A5	Return slot and True response to requestor					✓	
A6	Return False response to requestor					✓	
A7	Return error response to requestor				✓		✓
D I S P O S I T I O N				C5	C5	C5	C5
				Done	Done	Done	Done

Fig. 91 – Process Query Register 944

950 Load Request Processor		Load	Un load	Query Load	Query Frm	Oth		
C1	Request type?							
A1	Process Load (951)	✓						
A2	Process Unload (952)		✓					
A3	Process Query Load (953)			✓				
A4	Process Query Frames Loaded (954)				✓			
A5	Return error response to requestor					✓		
D I S P O S I T I O N		Done	Done	Done	Done	Done		

Fig. 92 – Load Request Processor 950

951		Load Request Processor Process Load							
C1	Is target clip registered?		Yes		No				
C2	Is bandwidth hogging in effect?		Yes	No					
C3	Is requestor clip a bandwidth hog?	Yes	No						
C4	Any argument errors?				Yes		No		
C5	Do permissions allow load?					Yes		No	
C6	Requestor supplying destination instance?					No	Yes		
A1	Get arguments	✓		✓					
A2	Put request onto reschedule list		✓						
A3	Create local instance at specified depth on the specified level					✓			
A4	Pick target clip variant (if any) based on cycle and bandwidth environment (Cycle Category and Bandwidth)					✓	✓		
A5	Initiate target clip load					✓	✓		
A6	Add target clip to load list for Load Check (920)					✓	✓		
A7	Return OK response to requestor					✓	✓		
A8	Return error response to requestor				✓	✓			✓
	D I S P O S I T I O N		C4	Done	C4	Done	Done	Done	Done

Fig. 93 – Process Load 951

952		Load Request Processor Process Unload					
C1	Is clip loaded?		Yes	No			
C2	Do permissions allow load?		Yes	No			
C3	Is clip playing?	Yes	No				
A1	Stop clip play	✓					
A2	Propagate unload to clip's children that obey	✓	✓				
A3	Initiate target clip unload	✓	✓				
A4	Set clip state as unloaded	✓	✓				
A5	Return OK response to requestor	✓	✓				
A6	Return error response to requestor			✓	✓		
D I S P O S I T I O N		Done	Done	Done	Done		

Fig. 94 – Process Unload 952

953		Load Request Processor Process Query Load					
C1	Is clip loaded?		Yes	No			
A1	Return True response to requestor	✓					
A2	Return False response to requestor		✓				
D I S P O S I T I O N		Done	Done				

Fig. 95 – Process Query Load 953

954	Load Request Processor Process Query Load Frame							
C1	Is clip loading or loaded?	Yes	No					
A1	Return frames loaded count to requestor	✓						
A2	Return error response to requestor	✓						
	D I S P O S I T I O N	Done	Done					

Fig. 96 – Process Query Frames Loaded 954

960	Play Request Processor							
C1	Request type?	Play	Paus	Stop	Query	Query	Oth	
				Play	Frm			
A1	Process Play (961)	✓						
A2	Process Pause (962)		✓					
A3	Process Stop (963)			✓				
A4	Process Query Play (964)				✓			
A5	Process Query Frame (965)					✓		
A6	Return error response to requestor						✓	
	D I S P O S I T I O N	Done	Done	Done	Done	Done	Done	

Fig. 97 – Play Request Processor 960

961 Play Request Processor Process Play		Yes	No					
C1	Is clip loaded?							
C2	Do permissions allow play?	Yes	No					
A1	Call clip's play routine	✓						
A2	Propagate play to clip's children that obey	✓						
A3	Set clip state as playing	✓						
A4	Return OK response to requestor	✓						
A5	Return error response to requestor		✓	✓				
D I S P O S I T I O N		Done	Done	Done				

Fig. 98 – Process Play 961

962 Play Request Processor Process Pause		Yes	No					
C1	Is clip playing?							
C2	Do permissions allow pause?	Yes	No					
A1	Call clip's pause routine	✓						
A2	Propagate pause to clip's children that obey	✓						
A3	Set clip state as paused	✓						
A4	Return OK response to requestor	✓						
A5	Return error response to requestor		✓	✓				
D I S P O S I T I O N		Done	Done	Done				

Fig. 99 – Process Pause 962

963		Play Request Processor Process Stop						
C1	Is clip playing or paused?		Yes	No				
C2	Do permissions allow stop?	Yes	No					
A1	Call clip's stop routine	✓						
A2	Propagate stop to clip's children that obey	✓						
A3	Set clip state as stopped	✓						
A4	Return OK response to requestor	✓						
A5	Return error response to requestor		✓	✓				
		D I S P O S I T I O N			Done	Done	Done	

Fig. 100 – Process Stop 963

964		Play Request Processor Process Query Play						
C1	Is clip playing?	Yes	No					
A1	Return True response to requestor	✓						
A2	Return False response to requestor		✓					
		D I S P O S I T I O N			Done	Done		

Fig. 101 – Process Query Play 964

965 Play Request Processor		Process Query Play Frame					
C1	Is clip playing, paused or stopped?	Yes	No				
A1	Return current play frame to requestor	✓					
A2	Return OK response to requestor	✓					
A3	Return zero as play frame to requestor		✓				
A4	Return error response to requestor		✓				
D I S P O S I T I O N		Done	Done				

Fig. 102 – Process Query Play Frame 965

970 Position Request Processor							
C1	Request type?	FF	Re wind	Query	Wind Rew	Oth	
A1	Process Fast Forward (971)	✓					
A2	Process Rewind (972)		✓				
A3	Process Query Rewound (973)			✓			
A4	Return error response to requestor				✓		
D I S P O S I T I O N		Done	Done	Done	Done		

Fig. 103 – Position Request Processor 970

971	Position Request Processor Process Fast Forward						
C1	Is clip loaded?		Yes	No			
C2	Do permissions allow fast forward?	Yes	No				
A1	Call clip's fast forward routine	✓					
A2	Propagate fast forward to clip's children that obey	✓					
A3	Return OK response to requestor	✓					
A4	Return error response to requestor		✓	✓			
	D I S P O S I T I O N	Done	Done	Done			

Fig. 104 – Process Fast Forward 971

972	Position Request Processor Process Rewind						
C1	Is clip loaded?		Yes	No			
C2	Do permissions allow rewind?	Yes	No				
A1	Call clip's rewind routine	✓					
A2	Propagate rewind to clip's children that obey	✓					
A3	Return OK response to requestor	✓					
A4	Return error response to requestor		✓	✓			
	D I S P O S I T I O N	Done	Done	Done			

Fig. 105 – Process Rewind 972

973	Position Request Processor Process Query Rewound								
C1	Is clip at first frame?	Yes	No						
A1	Return True response to requestor	✓							
A2	Return False response to requestor		✓						
	D I S P O S I T I O N	Done	Done						

Fig. 106 – Process Query Rewound 973

987	State Request Processor								
C1	Request type?	Cyc Pig On	Cyc Pig Off	BW Hog On	BW Hog Off	Query State	Qry Cyc	Qry BW	Other
A1	Process Cycle Pig On (981)	✓							
A2	Process Cycle Pig Off (982)		✓						
A3	Process Bandwidth Hog On (983)			✓					
A4	Process Bandwidth Hog Off (984)				✓				
A5	Process Query State (985)					✓			
A5	Process Query Cycle Pig (986)						✓		
A5	Process Query Bandwidth Hog (987)							✓	
A6	Return error response to requestor								✓
	D I S P O S I T I O N	Done	Done	Done	Done	Done	Done	Done	Done

Fig. 107 – State Request Processor 980

981	State Request Processor Process Cycle Pig On						
C1	Is clip loaded?	Yes	No				
C2	Any argument errors?			Yes	No		
C3	Are sufficient cycles available?				Yes	No	
A1	Get arguments	✓					
A2	Deduct clip requirements from cycles available			✓			
A3	Add clip to intensive task list			✓			
A4	Return OK response to requestor			✓			
A5	Return error response to requestor		✓	✓	✓		
	D I S P O S I T I O N	C2	Done	Done	Done	Done	

Fig. 108 – Process Cycle Pig On 981

982	State Request Processor Process Cycle Pig Off						
C1	Is clip loaded?	Yes	No				
C3	Is clip on intensive task list?	Yes	No				
A1	Remove clip from intensive task list	✓					
A2	Restore clip requirements to cycles available	✓					
A3	Return OK response to requestor	✓					
A4	Return error response to requestor		✓	✓			
	D I S P O S I T I O N	Done	Done	Done			

Fig. 109 – Process Cycle Pig Off 982

983		State Request Processor Process Bandwidth Hog On						
C1	Is clip loaded?		Yes	No				
C2	Any argument errors?			Yes	No			
C3	Is sufficient bandwidth available?				Yes	No		
A1	Get arguments	✓						
A2	Deduct clip requirements from bandwidth available			✓				
A3	Add clip to high bandwidth task list			✓				
A4	Return OK response to requestor			✓				
A5	Return error response to requestor		✓	✓	✓			
D I S P O S I T I O N			C2	Done	Done	Done	Done	

Fig. 110 – Process Bandwidth Hog On 983

984		State Request Processor Process Bandwidth Hog Off						
C1	Is clip loaded?		Yes	No				
C3	Is clip on high bandwidth list?	Yes	No					
A1	Remove clip from high bandwidth task list	✓						
A2	Restore clip requirements to bandwidth available	✓						
A3	Return OK response to requestor	✓						
A4	Return error response to requestor		✓	✓				
D I S P O S I T I O N			Done	Done	Done			

Fig. 111 – Process Bandwidth Hog Off 984

985 State Request Processor Process Query State											
C1	Is clip loaded?	Yes	No								
C2	Is clip on intensive task list?			Yes	No						
C3	Is clip on high bandwidth task list?					Yes	No				
A1	Return clip state to requestor	✓									
A2	Return Intensive Task True response to requestor		✓								
A3	Return Intensive Task False response to requestor			✓							
A4	Return High Bandwidth Task True response to requestor				✓						
A5	Return High Bandwidth Task False response to requestor					✓					
A6	Return OK response to requestor					✓	✓				
A7	Return error response to requestor	✓									
D I S P O S I T I O N		C2	Done	C3	C3	Done	Done				

Fig. 112 – Process Query State 985

986 State Request Processor Process Query Cycle Pig											
C1	Is any clip on intensive task list?	Yes	No								
A1	Return True response to requestor	✓									
A2	Return cycles available to requestor	✓									
A3	Return False response to requestor		✓								
	D I S P O S I T I O N	Done	Done								

Fig. 113 – Process Query Cycle Pig 986

987	State Request Processor Process Query Bandwidth Hog						
C1	Is any clip on high bandwidth task list?	Yes	No				
A1	Return True response to requestor	✓					
A2	Return bandwidth available to requestor	✓					
A3	Return False response to requestor		✓				
	D I S P O S I T I O N	Done	Done				

Fig. 114 – Process Query Bandwidth Hog 987

990	Local Volume Request Processor						
C1	Request type?	Set Loc Vol	Loc Vol On	Loc Vol Off	Query Loc Vol	Oth	
A1	Process Set Local Volume (991)	✓					
A2	Process Turn Local Volume On (992)		✓				
A3	Process Turn Local Volume Off (993)			✓			
A4	Process Query Local Volume (994)				✓		
A5	Return error response to requestor					✓	
	D I S P O S I T I O N	Done	Done	Done	Done	Done	

Fig. 115 – Local Volume Request Processor 990

991 Local Volume Request Processor Process Set Local Volume	
C1	Is clip loaded?
	Yes Yes
C2	Do permissions allow local volume control?
	Yes Yes
C3	Is Global Volume ON?
	Yes Yes
C4	Is Local Volume ON?
	Yes Yes
A1	Get volume argument ✓
A2	Call clip's set sound volume routine ✓
A3	Save sound volume in clip's slot ✓ ✓ ✓
A5	Return OK response to requestor ✓ ✓ ✓
A6	Return error response to requestor ✓ ✓
D I S P O S I T I O N	
	C3 Done Done Done Done Done

Fig. 116 – Process Set Local Volume 991

992 Local Volume Request Processor Process Turn Local Volume On	
C1	Is clip loaded?
	Yes Yes
C2	Do permissions allow local volume control?
	Yes Yes
C3	Is Global Volume ON?
	Yes Yes
A1	Call clip's set sound volume routine ✓
A2	Set volume ON in clip's slot ✓ ✓
A3	Propagate volume ON to clip's children that obey ✓ ✓
A4	Return OK response to requestor ✓ ✓
A5	Return error response to requestor ✓ ✓
D I S P O S I T I O N	
	Done Done Done Done

Fig. 117 – Process Local Volume On 992

993 Local Volume Request Processor Process Turn Local Volume Off					
C1	Is clip loaded?		Yes	No	
C2	Do permissions allow local volume control?		Yes	No	
C3	Is Global Volume ON?	Yes	No		
A1	Call clip's set sound volume routine	✓			
A2	Set volume OFF in clip's slot	✓	✓		
A3	Propagate volume OFF to clip's children that obey	✓	✓		
A4	Return OK response to requestor	✓	✓		
A5	Return error response to requestor			✓	✓
D I S P O S I T I O N		Done	Done	Done	Done

Fig. 118 – Process Local Volume Off 993

994 Local Volume Request Processor Process Query Local Volume					
C1	Is clip loaded?	Yes	No		
A1	Return clip's local sound volume to requestor	✓			
A2	Return clip's on/off status to requestor	✓			
A3	Return OK response to requestor	✓			
A4	Return error response to requestor		✓		
D I S P O S I T I O N		Done	Done		

Fig. 119 – Process Query Local Volume 994

1000 Global Volume Request Processor		Set Gbl Vol	Gbl Vol On	Gbl Vol Off	Query Gbl Vol	Oth		
C1	Request type?							
A1	Process Set Global Volume (1001)	✓						
A2	Process Turn Global Volume On (1002)		✓					
A3	Process Turn Global Volume Off (1003)			✓				
A4	Process Query Global Volume (1004)				✓			
A5	Return error response to requestor					✓		
D I S P O S I T I O N		Done	Done	Done	Done	Done		

Fig. 120 – Global Volume Request Processor 1000

1001 Global Volume Request Processor		Process Set Global Volume	Yes	No				
C1	Do permissions allow global volume control?							
C2	Is Global Volume ON?			Yes	No			
A1	Get volume argument and convert to the fraction of the maximum volume (between 0.0 and 1.0)	✓						
A2	Save global volume and fraction	✓						
A3	Process Turn Global Volume On (1002) to invoke new volume level		✓					
A4	Return OK response to requestor			✓	✓			
A5	Return error response to requestor	✓						
D I S P O S I T I O N		C2	Done	Done	Done			

Fig. 121 – Process Set Global Volume 1001

1002	Global Volume Request Processor Process Turn Global Volume On							
C1	Do permissions allow global volume control?	Yes	No					
L2	Loop through all clips		Ent	Loop			Exit	
C3	More clips?			Yes	No			
C4	Is clip local volume ON?					Yes	No	
A1	Set global sound volume ON	✓						
A2	Get next clip			✓				
A3	Call clip's set sound volume routine (with clip's volume level argument)					✓		
A4	Return OK response to requestor							✓
A5	Return error response to requestor	✓						
	D I S P O S I T I O N	L2 Ent	L2 Done	C4 Loop	L2 Exit	L2 Loop	L2 Loop	L2 Done

Fig. 122 – Process Global Volume On 1002

1003	Global Volume Request Processor Process Turn Global Volume Off							
C1	Do permissions allow global volume control?	Yes	No					
L2	Loop through all clips		Ent	Loop		Loop		Exit
C3	More clips?			Yes	No			
C4	Is clip local volume ON?					Yes	No	
A1	Set global sound volume OFF	✓						
A2	Get next clip			✓				
A3	Call clip's set sound volume routine (with zero argument)					✓		
A4	Return OK response to requestor							✓
A5	Return error response to requestor	✓						
	D I S P O S I T I O N	L2 Ent	L2 Done	C4 Loop	L2 Exit	L2 Loop	L2 Loop	L2 Done

Fig. 123 – Process Global Volume Off 1003

1004	Global Volume Request Processor Process Query Global Volume						
C1	Global Volume?	ON	OFF				
A1	Return global sound ON status to requestor	✓					
A2	Return global sound OFF status to requestor		✓				
A3	Return global volume level to requestor	✓	✓				
A4	Return OK response to requestor	✓	✓				
	D I S P O S I T I O N	Done	Done				

Fig. 124 – Process Query Global Volume 1004

1100	Load Asset Manager Components						
C1	Enter	Yes					
A1	Process Dup List (1101)	✓					
A2	Load Network Bandwidth Speedometer (1102)	✓					
A3	Process Level List (1103)	✓					
A4	Process Swf List (1104)	✓					
A5	Process Speedo List (1105)	✓					
	D I S P O S I T I O N	Done					

Fig. 125 – Load Asset Manager Components 1100

1101	Load Asset Manager Components Process Dup List							
L1	Loop over Dup List entries	Ent	Loop	Exit				
C2	More entries?	Yes	No					
A1	Extract info about container	✓						
A2	Create Container (1106)	✓						
	D I S P O S I T I O N	L1 Loop	L1 Loop	L1 Exit	Done			

Fig. 126 – Process Dup List 1101

1102	Load Asset Manager Components Load Network Bandwidth Speedometer							
C1	Enter	Yes						
C2	Is load finished?		No	Yes				
A1	Extract Network Bandwidth Speedometer from 1st entry of Speedo List	✓						
A2	Create Container (1106)	✓						
A3	Load Clip (1107)	✓						
A4	Start Network Bandwidth Speedometer (1120)	✓						
	D I S P O S I T I O N	C2	C2	Done				

Fig. 127 – Load Network Bandwidth Speedometer 1102

1103 Load Asset Manager Components Process Level List						
L1	Loop over Level List entries	Ent	Loop	Exit		
C2	More entries?		Yes	No		
A1	Extract info about level		✓			
A2	Create Container (1106)		✓			
A3	Load Clip (1107)		✓			
D I S P O S I T I O N		L1	L1	L1		
		Loop	Loop	Exit	Done	

Fig. 128 – Process Level List 1103

1104 Load Asset Manager Components Process Swf List						
L1	Loop over Swf List entries	Ent	Loop	Exit		
C2	More entries?		Yes	No		
A1	Extract info about .swf file		✓			
A2	Create Container (1106)		✓			
A3	Load Clip (1107)		✓			
D I S P O S I T I O N		L1	L1	L1		
		Loop	Loop	Exit	Done	

Fig. 129 – Process Swf List 1104

1105	Load Asset Manager Components Process Speedo List						
L1	Loop over Speedo List entries	Ent	Loop	Exit			
C2	More entries?	Yes	No				
A1	Extract info about speedometer or spy file	✓					
A2	Create Container (1106)	✓					
A3	Load Clip (1107)	✓					
	D I S P O S I T I O N	L1 Loop	L1 Loop	L1 Exit	Done		

Fig. 130 – Process Speedo List 1105

1106	Load Asset Manager Components Create Container						
C1	Is load at a level?	Yes	No				
C2	Is a container supplied?	Yes	No				
C3	Are coordinates supplied?			Yes	No		
A1	Set up level args for Load Clip	✓					
A2	Create container with name and depth on level specified in args		✓				
A3	Set coordinates			✓			
A4	Set up container args for Load Clip			✓	✓		
	D I S P O S I T I O N	Done	C3	C3	Done	Done	

Fig. 131 – Create Container 1106

1107	Load Asset Manager Components Load Clip								
C1	Is Network Bandwidth Speedometer to be notified of this load?	Yes	No						
C2	Is clip load to be monitored?		Yes	No					
A1	Force monitor target to clip Playing	✓							
A2	Add clip to monitor list with start time and load state of Initied		✓						
A3	Start Load Monitor (1110) (may already be running--and that's		✓						
A4	Load clip into level/container specified		✓	✓					
	D I S P O S I T I O N	C2	C2	Done	Done				

Fig. 132 – Load Clip 1107

1110	Load Monitor								
C1	Are there entries in the monitor list?	No		Yes					
L2	Loop through loads to be monitored	Ent		Loop	Exit				
C3	More entries to process?		Yes	No					
C4	Has load reached monitor target?	Yes	No						
A1	Stop until restarted	✓							
A2	Process Monitor State (1111)		✓						
A3	Wait until next frame				✓				
	D I S P O S I T I O N	Stop C1	L2 Loop	L2 Loop	L2 Loop	L2 Exit	Wait C1		

Fig. 133 – Load Monitor 1110

1111 Load Monitor Process Monitor State		Init ed	Load ing	Load ed	Play ing	Stop ped	Done		
C1	Current monitor state?								
C2	Has monitor target been reached?							Yes	No
A1	Check Loading Started (1112)				✓				
A2	Check Loading Complete (1113)				✓				
A3	Check Playing Started (1114)				✓				
A4	Check Playing Stopped (1115)				✓				
A5	Call callback routine, if any							✓	
A6	Mark entry done							✓	
D I S P O S I T I O N		C3	C3	C3	C3	Done	Done	Done	Done

Fig. 134 – Process Monitor State 1111

1112 Load Monitor Check Loading Started		Yes	No						
C1	Is current load frame greater than 0?	Yes	No						
C2	Has timeout expired?		No	Yes					
A1	Set monitor state to Loading	✓							
A2	Increment time for timeout test		✓						
A3	Set load state to Done			✓					
D I S P O S I T I O N		Done	Done	Done					

Fig. 135 – Check Loading Started 1112

1113		Load Monitor Check Loading Complete							
C1	Is current load frame at total frames?	Yes	No						
C2	Has timeout expired?		No	Yes					
A1	Set monitor state to Loaded	✓							
A2	Increment time for timeout test		✓						
A3	Set load state to Done			✓					
		D I S P O S I T I O N				Done	Done	Done	

Fig. 136 – Check Loading Compete 1113

1114		Load Monitor Check Playing Started							
C1	Is current frame greater than 0?	Yes	No						
C2	Has timeout expired?		No	Yes					
C3	Is Network Bandwidth Speedo to be informed?				Yes	No			
A1	Set monitor state to Playing	✓							
A2	Increment time for timeout test		✓						
A3	Set load state to Done			✓					
A4	Report clip to Network Bandwidth Speedo (1120)				✓				
		D I S P O S I T I O N				C3	Done	Done	Done

Fig. 137 – Check Playing Started 1114

1115	Load Monitor Check Playing Stopped						
C1	Is current frame same as last check?	Yes	No				
C2	Has timeout expired?		No Yes				
A1	Set monitor state to Stopped	✓					
A2	Increment time for timeout test		✓				
A3	Set load state to Done			✓			
	D I S P O S I T I O N	Done	Done	Done			

Fig. 138 – Check Playing Stopped 1115

1120	Network Bandwidth Speedometer						
C1	Is speedo on?	No	Yes				
C2	Has average changed bandwidth category?			Yes	No		
A1	Get current time as clip's finish time	✓					
A2	Compute load time from start and finish	✓					
A3	Compute instantaneous rate for that clip (in bytes per second)	✓					
A4	Accumulate load time and size into a new running average rate (in bytes)	✓					
A5	Set new Bandwidth Category (used for clip variant choice (see			✓			
A6	Set new Bandwidth Available value (used for Bandwidth Hog processing)			✓			
	D I S P O S I T I O N	Done	C2	Done	Done		

Fig. 139 – Network Bandwidth Speedometer 1120

1121	Network Bandwidth Speedometer Turn On Speedo						
C1	Is speedo on?	No	Yes				
A1	Set ON flag	✓					
	D I S P O S I T I O N	Done	Done				

Fig. 140 – Turn On Speedo 1121

1122	Network Bandwidth Speedometer Turn Off Speedo							
C1	Is speedo on?	No	Yes					
A1	Clear ON flag		✓					
	D I S P O S I T I O N	Done	Done					

Fig. 141 – Turn Off Speedo 1122

1130	CPU Cycles Speedometer							
C1	Enter at frame 1		Yes					
C2	Is speedometer on?	No	Yes					
C3	Has average changed cycles category?			Yes	No			
C4	Has control variable frame number been reached?					No	Yes	
C5	Is this the last frame?					No	Yes	
A1	Get start time		✓					
A2	Run measured loop		✓					
A3	Get finish time		✓					
A4	Compute loop time from start and finish		✓					
A5	Compute instantaneous rate for loop (in operations per second)		✓					
A6	Accumulate loop time and size into a new running average rate (in ops per		✓					
A7	Set new Cycles Category (used for clip variant choice (see			✓				
A8	Set new Cycles Available value (used for Cycle Pig processing)			✓				
A9	Set new control variable			✓				
A10	Advance to next frame			✓	✓	✓		
A11	Jump back to frame 1						✓	✓
	D I S P O S I T I O N	Stop	C1	C3	C4	C4	C4	C1

Fig. 142 – CPU Cycles Speedometer 1130

1131	CPU Cycles Speedometer Turn On Speedo	Yes	No				
C1	Is speedometer on?						
A1	Set loop enable flag			✓			
A2	Start loop (1130)			✓			
D I S P O S I T I O N						Done	Done

Fig. 143 – Turn On Speedo 1131

1132	CPU Cycles Speedometer Turn Off Speedo	Yes	No				
C1	Is speedometer on?						
A1	Clear loop enable flag			✓			
D I S P O S I T I O N						Done	Done

Fig. 144 – Turn Off Speedo 1132

1140	Frame Rate Speedometer						
C1	Enter at frame 1	Yes					
C2	Is speedometer on?	No	Yes				
C3	Does frame number equal control variable?			No	Yes		
C4	Has average changed frame rate category?					Yes	No
A1	Get start time	✓					
A2	Advance to next frame	✓	✓				
A3	Get finish time			✓			
A4	Compute frame time from start and finish			✓			
A5	Compute instantaneous frame rate (in frames per second)			✓			
A6	Accumulate frame time and count into a new running average rate (in frames)			✓			
A7	Set new Frame Rate Category				✓		
A8	Adjust Cycles Category and Cycles Available for Frame Rate Category (used for clip variant choice (see				✓		
A9	Jump back to frame 1					✓	✓
	D I S P O S I T I O N		Stop				
		C1	C3	C3	C4	C1	C1

Fig. 145 – Frame Rate Speedometer 1140

1141	Frame Rate Speedometer Turn On Speedo						
C1	Is speedometer on?	Yes	No				
A1	Set loop enable flag	✓					
A2	Start loop (1140)	✓					
	D I S P O S I T I O N		Done	Done			

Fig. 146 – Turn On Speedo 1141

1142 Frame Rate Speedometer Turn Off Speedo		Yes	No						
C1	Is speedometer on?								
A1	Clear loop enable flag	<input checked="" type="checkbox"/>							
D I S P O S I T I O N		Done	Done						

Fig. 147 – Turn Off Speedo 1142

1200	Bootstrap RunTime Bootstrap						
C1	Starting?	Yes					
A1	Site URL access causes bootstrap to be loaded Bootstrap performs remaining operations	✓					
A2	Set startup flag	✓					
A3	Load Preloader (1210)	✓					
A4	Load Asset Manager/Active Content Loader (1220)	✓					
A5	Load Session (1230)	✓					
A6	Load Tables (1240)	✓					
DISPOSITION		Done					

Fig. 148 – Bootstrap 1200

1210	Load Preloader RunTime Bootstrap						
C1	Entry?	Yes					
C2	Preloader loaded?		No	Yes			
C3	Was preloader load fast or slow?				Fast	Slow	
A1	Load Preloader Get load start time	✓					
A2	Get load finish time Compute total load time Save load time and size for later			✓			
A3	Preloader starts playing automatically					✓	
A4	Preloader sets its pointer in the /vars block					✓	
DISPOSITION		C2	C2	C3	Done	Done	

Fig. 149 – Load Preloader 1210

1220	Load Asset Manager / Active Content Loader RunTime Bootstrap						
C1	Entry?	Yes					
C2	AM / ACL loaded?		No	Yes			
A1	Load AM / ACL Get load start time	✓					
A2	Get load finish time Compute total load time Save load time and size for later		✓				
A3	AM initializes itself		✓				
A4	AM sets its /vars block pointer to indicate it is initialized and ready to operate		✓				
A5	ACL initializes itself		✓				
A6	ACL sets its /vars block pointer to indicate it is initialized and ready to operate		✓				
DISPOSITION		C2	C2	Done			

Fig. 150 – Load Asset Manager / Active Content Loader 1220

1230	Load Session RunTime Bootstrap						
C1	Entry?	Yes					
C2	Is Session loaded?		No	Yes			
A1	Register with AM for Session Save Session Slot pointer	✓					
A2	Set Session info into its slot Mark slot urgent	✓					
A3	Issue load request to AM via slot	✓					
A4	Session starts playing automatically (1300)		✓				
DISPOSITION		C2	C2	Done			

Fig. 151 – Load Session 1230

1240	Load Tables RunTime Bootstrap	Yes						
C1	Entry?	Yes						
C2	Is startup flag still set?	Yes	No					
C3	Are Tables loaded?			No	Yes			
A1	Register with AM for Tables Save Tables Slot pointer	✓						
A2	Set Tables info into its slot Mark slot normal		✓					
A3	Issue load request to AM via slot		✓					
A4	Tables starts playing automatically				✓			
A5	Tables sets its /vars block pointer to indicate it is initialized and ready to operate					✓		
	DISPOSITION	C2	C2	C3	C3	Done		

Fig. 152 – Load Tables 1240

1300 Init Projects Session Operation		No	Yes	Ent	Loop	Exit				
C1	Is Session completely loaded?									
L2	For each project			Ent	Loop	Exit				
C3	More projects?					Yes	No			
C4	Is first project loaded?							No	Yes	
A1	Session starts playing automatically	✓	✓							
A2	Set ready flag		✓							
A3	Set pointer for base of persistent data blocks		✓							
A4	Dup container for project file Set project pointer array in /vars block				✓					
A5	Register with AM for first project Save project Slot pointer							✓		
A6	Set first project info into its slot Mark slot urgent							✓		
A7	Issue load request to AM via slot Set current project index to 1							✓		
A8	First Project starts playing automatically (1400)									✓
A9	Init Projects - continued (1310)									✓
DISPOSITION		C1	L2	L2 Loop	L2 Loop	L2 Exit	C4	C4	Done	

Fig. 153 – Play Session 1300

1310 Init Projects - continued Session Operation		No	Yes	Ent	Loop	Exit				
L1	For each remaining project after the first			Ent	Loop	Exit				
C3	More projects?		Yes	No						
A1	Register with AM for project Save project Slot pointer		✓							
A2	Set project info into its slot Mark slot urgent		✓							
DISPOSITION		L1	L1	L1 Loop	Exit	Done				

Fig. 154 – Init Projects 1310

1320	doDone() Session Routine						
C1	Entry?	Yes					
C2	Project index valid?	Yes	No				
A1	Increment current project index	✓					
A2	Select slot for new current project	✓					
A3	Issue load request to AM via slot	✓					
A4	Project starts playing automatically (1400)	✓					
A5	Web site finished	✓					
	DISPOSITION	C2	Done	Done			

Fig. 155 – Session doDone() 1320

1400	Play Project Project Operation							
C1	Is Project completely loaded?	No	Yes					
A1	Project starts playing automatically	✓	✓					
A2	Set ready flag		✓					
A3	Initialize project variables Set current project pointer in /vars block		✓					
A4	Dup container for Scene 1		✓					
A5	Register with AM for Scene 1 Save project Slot pointer		✓					
A6	Set Scene 1 info into its slot Mark slot urgent		✓					
A7	Issue prep, stage, load and play requests to AM via slot Set current scene index to 1		✓					
A8	Dup container for Scene 2		✓					
A9	Register with AM for Scene 2 Save project Slot pointer		✓					
A10	Set Scene 2 info into its slot Mark slot normal		✓					
A11	Issue prep and stage requests to AM via slot		✓					
A12	Wait Scene 1 Play (1410)		✓					
	DISPOSITION	C1	Done					

Fig. 156 – Play Project 1400

1410 Wait Scene 1 Play Project Operation		No	Yes					
C1	Is Scene 1 playing?			No	Yes			
C2	Are Tables ready?			No	Yes			
A1	Clear startup flag		✓					
A2	Request new table for Scene Table Save table pointer			✓				
A3	Request new table for Scene 2 Quicklink Table Save table pointer in Scene 2 slot			✓				
A4	Init Scene Table			✓				
A5	Init Scene 2 Quicklink Table			✓				
A6	Request new table for Scene 1 Quicklink Table Save table pointer in Scene 1 slot			✓				
A7	Init Scene 1 Quicklink Table			✓				
A8	Add all quicklinkable Scene 1 components to the Scene 1 Quicklink Table			✓				
A9	Set current Quicklink Table pointer to Scene 1's Quicklink Table			✓				
A10	Build Scene Table (1420)			✓				
DISPOSITION		C1	C2	C2	Done			

Fig. 157 – Wait Scene 1 Play 1410

1420	Build Scene Table Project Operation							
C1	Entry?	Yes						
L2	Create remaining scenes		Entry	Loop	Exit			
C3	More scenes?			Yes	No			
A1	Add Scene 1 to Scene Table	✓						
A2	Add Scene 2 to Scene Table	✓						
A3	Duplicate container for scene		✓					
A4	Register scene with AM Save scene slot pointer		✓					
A5	Set scene info in slot Mark slot normal		✓					
A6	Request new table for scene's Quicklink Table		✓					
A7	Add scene to Scene Table		✓					
A8	Finish Scene Init (1430)					✓		
	DISPOSITION	L2 Entry	L2 Loop	L2 Loop	L2 Exit	Done		

Fig. 158 – Build Scene Table 1420

1430	Finish Scene Init Project Operation							
L1	For scenes after Scene 2		Entry	Loop	Exit			
C2	More scenes?		Yes	No				
A1	Init Scene's Quicklink Table		✓					
A2	Set project table pointer to indicate that project tables are available				✓			
	DISPOSITION	L1 Loop	L1 Loop	L1 Exit	Done			

Fig. 159 – Finish Scene Init 1430

1440 doDone() Project Routine		No	Yes						
C1	Are tables available?								
C2	Looping enabled?			Yes	No				
C3	Scene index too large?			Yes	No				
A1	Ignore request	✓							
A2	Call doOldScene() (1490)		✓						
A3	Increment current scene index		✓						
A4	Reset current scene index to 1			✓					
A5	Call doNewScene() (1500)			✓	✓	✓			
A6	Call doNextScene() (1510)			✓	✓	✓			
DISPOSITION		Done	C2	Done	Done	Done			

Fig. 160 – Project doDone() 1440

1450	doJump() Project Routine							
C1	Are tables available?		No	Yes				
C2	Scene Found?				No	Yes		
A1	Ignore request	✓						
A2	Extract destination scene ID	✓						
A3	Look up ID in Scene Table	✓						
A4	Call doOldScene() (1490)		✓	✓				
A5	Call Session doDone() (1310)		✓					
A6	Set new destination scene ID		✓					
A7	Mark new scene's slot urgent		✓					
A8	Issue prep, stage, and load requests to AM via slot		✓					
A9	Call doNewScene() (1500)		✓					
A10	Call doNextScene() (1510)		✓					
DISPOSITION			Done	C2	Done	Done		

Fig. 161 – Project doJump() 1450

1460	doForceDone() Project Routine							
C1	Entry?		Yes					
A1	Set force flag true	✓						
A2	Call doDone() (1440)	✓						
DISPOSITION			Done					

Fig. 162 – Project doForceDone() 1460

1470	doHold() Project Routine						
C1	Entry?		Yes				
A1	Increment hold count		✓				
	DISPOSITION			Done			

Fig. 163 – Project doHold() 1470

1480	doRelease() Project Routine						
C1	Entry?		Yes				
A1	Decrement hold count		✓				
	DISPOSITION						

Fig. 164 – Project doRelease() 1480

1490	doOldScene() Project Routine						
C1	Are tables available?		No	Yes			
C2	More than 1 scene?			No	Yes		
C3	More than 2 scenes?				No	Yes	
A1	Ignore request		✓				
A2	Save current scene index as old index			✓			
A3	Clear current Quicklink table pointer in /vars block				✓	✓	
A4	Retrieve scene entry in Scene Table				✓	✓	
A5	Call scene doEnd() (to hide scene)				✓	✓	
A6	Issue unload request to AM via scene's slot					✓	
	DISPOSITION		Done	C2	Done	C3	Done

Fig. 165 – Project doOldScene() 1490

1500 doNewScene() Project Routine		No	Yes						
C1	Are tables available?								
C2	More than 1 scene?			No	Yes				
C3	More than 2 scenes?					No	Yes		
A1	Ignore request	✓							
A2	Retrieve scene entry in Scene Table		✓						
A3	Restart scene timeline			✓					
A4	Issue play request to AM via scene's slot				✓				
A5	Set current scene pointer in /vars Set current quicklink table pointer in /vars					✓	✓		
DISPOSITION		Done	Done	C3	Done	Done			

Fig. 166 – Project doNewScene() 1500

1510 doNextScene() Project Routine		Yes							
C1	Entry?								
C2	Is Scene ID valid?		Yes	No					
C3	Looping enabled?			No	Yes				
C4	More than 2 scenes?				No	Yes			
A1	Set next scene index to current index + 1	✓							
A2	Ignore request		✓						
A3	Reset scene index to 1			✓					
A4	Issue prep, stage, and load requests to AM via scene's slot					✓			
DISPOSITION		C2	C4	Done	C4	Done	Done		

Fig. 167 – Project doNextScene() 1510

1600	Play Scene Scene Operation								
C1	Is Scene completely loaded?	No	Yes						
A1	Scene starts playing automatically	✓	✓						
A2	Set ready flag		✓						
A3	Init scene time variables		✓						
A4	Initialize other scene variables		✓						
A5	Stop and wait for Begin Scene (1610)		✓						
	DISPOSITION	C1	Done						

Fig. 168 – Play Scene 1600

1610	Begin Scene Scene Operation								
C1	Play at frBegin?	No	Yes						
A1	Stopped, waiting for play	✓							
A2	Call doShow() on each component's control to turn it on		✓						
A3	Set scene start time to current timer time		✓						
A4	Init for (re)play of scene "Needed" Done count set to component count		✓						
A5	Wait for scene timeout (1620)		✓						
	DISPOSITION	C1	Done						

Fig. 169 – Begin Scene 1610

1620 Wait Scene Timeout Scene Operation		No	Yes						
C1	Has scene timeout elapsed?								
C2	Is scene being held?			Yes	No				
C3	Done flag?				Set	Clr			
C4	Is scene being forced done?						Yes	No	
A1	Set Done Flag		✓						
A2	Call project doRelease() (1480)				✓		✓		
A3	Call project doDone() (1440)				✓		✓		
A4	Clear Done Flag				✓		✓		
DISPOSITION		C2	C2	C4	Done	C4	Done	C1	

Fig. 170 – Wait Scene Timeout 1620

1630 doDone() Scene Routine		Yes							
C1	Entry?								
C2	Does "done" count equal or exceed "needed"		Yes	No					
A1	Increment "done" count	✓							
A2	Set Done Flag		✓						
DISPOSITION		C2	Done	Done					

Fig. 171 – Scene doDone() 1630

1640	doForceDone() Scene Routine						
C1	Entry?		Yes				
A1	Set "force" flag		✓				
A2	Call project doRelease() (1480) for each "hold"		✓				
	DISPOSITION		Done				

Fig. 172 – Scene doForceDone() 1640

1650	doHold() Scene Routine						
C1	Entry?		Yes				
A1	Increment hold count		✓				
A2	Call project doHold() (1470)		✓				
	DISPOSITION		Done				

Fig. 173 – Scene doHold() 1650

1660	doRelease() Scene Routine						
C1	Entry?		Yes				
A1	Decrement hold count		✓				
A2	Call project doRelease() (1480)		✓				
	DISPOSITION		Done				

Fig. 174 – Scene doRelease() 1660

1670	doJump() Scene Routine						
C1	Entry?		Yes				
A1	Pass Jump ID to project		✓				
A2	Call project doJump() (1450)		✓				
	DISPOSITION		Done				

Fig. 175 – Scene doJump() 1670

1680	doEnd() Scene Routine						
C1	Entry?		Yes				
A2	Call doHide() on each component's control to turn it off		✓				
	DISPOSITION		Done				

Fig. 176 – Scene doEnd() 1680

1700	Asset Manager (56KB bandwidth version) Initialization and Startup							
C1	Entry?	Yes						
C2	Is network bandwidth high?		Yes	No				
A1	Initialize Asset Manager	✓						
A2	Initialize Active Content Loader (1900)	✓						
A3	Initialize Urgent queue	✓						
A4	Initialize Normal queue	✓						
A5	Load high bandwidth Asset Manager		✓					
A6	Set AM pointer in /vars block to indicate ready to operate		✓	✓				
A7	Start Operation Processing Loop (1800)		✓	✓				
A8	Wait for User Request Calls (1710)		✓	✓				
DISPOSITION				C2	Done	Done		

Fig. 177 – AM Init and Startup 1700

1710	Asset Manager (56KB bandwidth version) User Request Calls							
C1	User request?	Reg	Prep	Stg	Load	Play	Unld	Other ne
A1	Process Registration request (1720)	✓						
A2	Process Scene Prep request (1730)		✓					
A3	Process Scene Stage request (1740)			✓				
A4	Process Load request (1750)				✓			
A5	Process Play request (1760)					✓		
A6	Process Unload request (1770)						✓	
A7	Set error response							✓
DISPOSITION				C1	C1	C1	C1	C1

Fig. 178 – AM User Request Calls 1710

1720	Asset Manager (56KB bandwidth version) Process Registration request	
C1	Entry?	Yes
A1	Create "slot" timeline	✓
A2	Initialize variables in slot	✓
A3	Initialize op list in slot	✓
A4	Return slot pointer	✓
	DISPOSITION	Done

Fig. 179 – AM Process Registration 1720

1730	Asset Manager (56KB bandwidth version) Process Scene Prep request	
C1	Entry?	Yes
A1	Set prep op and prep flag	✓
A2	Queue slot (1780)	✓
	DISPOSITION	Done

Fig. 180 – AM Scene Prep 1730

1740	Asset Manager (56KB bandwidth version) Process Scene Stage request	
C1	Entry?	Yes
A1	Set stage op and stage flag	✓
A2	Queue slot (1780)	✓
	DISPOSITION	Done

Fig. 181 – AM Scene Stage 1740

1750	Asset Manager (56KB bandwidth version) Process Load request							
C1	Entry?	Yes						
A1	Set load op and load flag	✓						
A2	Queue slot (1780)	✓						
	DISPOSITION	Done						

Fig. 182 - AM Scene Load 1750

1760	Asset Manager (56KB bandwidth version) Process Play request							
C1	Entry?	Yes						
A1	Set play op and play flag	✓						
A2	Queue slot (1780)	✓						
	DISPOSITION	Done						

Fig. 183 - AM Process Play 1760

1770	Asset Manager (56KB bandwidth version) Process Unload request							
C1	Entry?	Yes						
A1	Set unload op	✓						
A2	Queue slot (1780)	✓						
	DISPOSITION	Done						

Fig. 184 - AM Process Unload 1770

1780 Asset Manager (56KB bandwidth version) Queue slot													
C1	Entry?		Yes										
C2	Is slot's urgent flag set?			Yes	No								
A1	Add op to slot's op list		✓										
A2	Set slot busy		✓										
A3	Add slot to tail of Urgent queue			✓									
A4	Increment Urgent queue count			✓									
A5	Add slot to tail of Normal queue				✓								
A6	Increment Normal queue count				✓								
A7	Play Operation Processing Loop (1800)			✓	✓								
DISPOSITION		C2	Done	Done									

Fig. 185 – AM Queue Sot 1780

1790 Asset Manager (56KB bandwidth version) Dequeue slot													
C1	Is slot on urgent or normal queue?		urg	norm	oth								
A1	Remove slot from urgent queue		✓										
A2	Decrement urgent queue count		✓										
A3	Remove slot from normal queue			✓									
A4	Decrement normal queue count			✓									
A5	Set error				✓								
DISPOSITION		Done	Done	Done									

Fig. 186 – AM Dequeue Sot 1790

1800 Asset Manager (56KB bandwidth version) Operation Processing		Yes						
C1	Entry?							
C2	Does slot exist?		Yes	No				
C3	Does slot exist?				Yes	No		
C4	Is Urgent queue empty?				No	Yes		
A1	Select slot on Head of Urgent queue	✓					✓	
A2	Select slot on Head of Normal queue		✓					
A3	Process Op(s) in selected slot (1810)	✓		✓				
A4	Select next slot on Urgent queue	✓						
A5	Select next slot on Normal queue						✓	
NOTE	Code stops when both queues are empty. Code that adds a slot to a queue does a "play" which starts the code up again.					✓		
	DISPOSITION	C2	C2	C3	C4	Stop C1	C2	C3

Fig. 187 – AM Operation Processing 1800

1810	Asset Manager (56KB bandwidth version) Process Op(s) in selected slot							
C1	Is op index zero?	Yes	No					
C2	Is there an op?		Yes	No				
C3	Does response index match op index?		Yes	No				
C4	Is op complete?		Yes	No				
C5	Is op flag still set?					Yes	No	
A1	Initialize op index to one Initialize response index to zero	✓						
A2	Increment op index		✓					✓
A3	Select op (at op index)	✓	✓	✓				✓
A4	Set op flag				✓			
A5	Process op (1820)				✓			
A6	Set no such op error						✓	
A7	Set error response						✓	
A8	Reset op list					✓		
A9	Dequeue slot from its queue (1790)					✓	✓	
A10	Op still in progress			✓				
	DISPOSITION	C2	C2	C2	Done	C5	Done	C2

Fig. 188 – AM Process Op(s) In Slot 1810

1820 Asset Manager (56KB bandwidth version) Process Op		Prep	Stg	Load	Play	Unld	Oth		
C1	Op?								
A1	Clear op flag (indicates op exists)	✓	✓	✓	✓	✓			
A2	Set response index to op index	✓	✓	✓	✓	✓			
A3	Set busy response	✓	✓	✓	✓	✓			
A4	Process Prep operation (1830)	✓							
A5	Process Stage operation (1840)			✓					
A6	Process Load operation (1850)				✓				
A7	Process Play operation (1860)					✓			
A8	Process Unload operation (1870)						✓		
DISPOSITION		Done	Done	Done	Done	Done	Done		

Fig. 189 – AM Process Op 1820

1830 Asset Manager (56KB bandwidth version) Process Prep operation		Yes							
C1	Entry?	Yes							
A1	Set up scene file information	✓							
A2	Register with Active Content Loader for load (1920)	✓							
DISPOSITION		Done							

Fig. 190 – AM Process Prep Operation 1830

1840	Asset Manager (56KB bandwidth version) Process Stage operation								
C1	Entry?		Yes						
A1	Set up scene's data block files info		✓						
A2	Register with Active Content Loader for load (1920)		✓						
	DISPOSITION		Done						

Fig. 191 – AM Process Stage Operation 1840

1850	Asset Manager (56KB bandwidth version) Process Load operation								
C1	Scene or clip load?		scene	clip					
A1	Set up scene's control, chrome, and debug files info		✓						
A2	Set up clip's file info			✓					
A3	Register with Active Content Loader for load (1920)		✓	✓					
	DISPOSITION		Done	Done					

Fig. 192 – AM Process Load Operation 1850

1860	Asset Manager (56KB bandwidth version) Process Play operation								
C1	Scene or clip?		scene	clip					
A1	Play scene's frBegin frame		✓						
A2	Call clip's doPlay() function			✓					
	DISPOSITION		Done	Done					

Fig. 193 – AM Process Play Operation 1860

1870	Asset Manager (56KB bandwidth version) Process Unload operation							
C1	Scene or clip?	sc ene	clip					
A1	Call scene's doEnd() function		✓					
A1	Unload scene's control and chrome files		✓					
A2	Call clip's doHide() function			✓				
A2	Unload clip			✓				
	DISPOSITION	Done	Done					

Fig. 194 – AM Process Unload Operation 1870

1900 Active Content Loader Initialization and Startup	
C1	Entry?
	Yes
A1	Init ACL urgent and normal queues
A2	Init ACL working pointers
A3	Load Project Text file containing object parameters
A4	Init ACL bandwidth algorithm variables
A5	Init ACL CPU algorithm variables
A6	Start CPU speedometer (2070)
A7	Start frame rate speedometer (2080)
A8	Set ACL pointer in /vars block to indicate ready to operate
A9	Start Loader frame loop (1960)
A10	Wait for Asset Manager Request Calls (1910)
DISPOSITION	
	Done

Fig. 195 - ACL Init and Startup 1900

1910 Active Content Loader Asset Manager Request Calls	
C1	Request type?
	Reg Un reg
A1	Process Register (1920)
A2	Process Unregister (1930)
DISPOSITION	
	Done Done

Fig. 196 – ACL AM Request Calls 1910

1920 Active Content Loader Process Register		Prep	Stg	Load	Oth				
C2	Operation?								
C3	Clip or scene?			clip	sc ene				
A1	Set up for load of scene file	✓							
A2	Set up for load of scene data block file(s)		✓						
A3	Set up for load of clip file			✓					
A4	Set up for load of scene control block, chrome, and debug files				✓				
A5	Queue Slot (1940)	✓	✓	✓	✓				
A6	Set error					✓			
DISPOSITION		Done	Done	Done	Done	Done			

Fig. 197 – ACL Process Register 1920

1930 Active Content Loader Process Unregister		Yes	No						
C1	Is slot registered?								
A1	Dequeue slot (1950)	✓							
A3	Abort loads in progress (1990)	✓							
A4	Set error		✓						
DISPOSITION		Done	Done						

Fig. 198 – ACL Process Unregister 1930

1940 Active Content Loader Queue slot		Yes	No						
C1	Is slot's urgent flag set?								
A1	Add slot to tail of Urgent queue	✓							
A4	Increment Urgent queue count	✓							
A3	Add slot to tail of Normal queue		✓						
A4	Increment Normal queue count		✓						
A5	Play Loader Frame Loop (1960)	✓	✓						
DISPOSITION		Done	Done						

Fig. 199 – ACL Queue Slot 1940

1950 Active Content Loader Dequeue slot		urg	norm	oth					
C1	Is slot on urgent or normal queue?								
A1	Remove slot from urgent queue	✓							
A2	Decrement urgent queue count	✓							
A3	Remove slot from normal queue		✓						
A4	Decrement normal queue count		✓						
A5	Set error			✓					
DISPOSITION		Done	Done	Done					

Fig. 200 – ACL Requeue Slot 1950

1960		Active Content Loader Loader frame loop							
C1	Entry?		Yes						
C2	Does slot exist?		Yes	No					
C3	Does slot exist?			Yes	No				
C4	Is Urgent queue empty?				No	Yes			
A1	Set abort flag in slot from Normal queue						✓		
A2	Select slot on Head of Urgent queue	✓					✓		
A3	Select slot on Head of Normal queue			✓					
A4	Perform load processing (1970)		✓		✓				
A5	Select next slot on Urgent queue	✓							
A6	Select next slot on Normal queue							✓	
NOTE	Code stops when both queues are empty. Code that adds a slot to a queue does a "play" which starts the code up again.						✓		
	DISPOSITION		C2	C2	C3	C4	Stop C1	C2	C3

Fig. 201 – ACL Loader Frame Loop 1960

1970	Active Content Loader Perform load processing	Yes						
C1	Entry?	Yes						
C2	Is load finished??		Yes	No				
C3	Is load being aborted?			Yes	No			
A1	Check loads done (1980)	✓						
A2	Abort loads (1990)			✓				
A3	Set load done response		✓					
A4	Dequeue slot (1950)		✓	✓				
A5	Continue load processing (2000)				✓			
DISPOSITION		C2	Done	Done	Done			

Fig. 202 – ACL Perform Load Processing 1970

1980	Active Content Loader Check loads done	Ent	Loop	Exit				
L1	Loop thru loads in progress							
C2	Is there a load in progress to check?		Y	N				
C3	Is load in progress completed?		Y	N				
A1	Select 1st load in progress	✓						
A2	Mark component loaded		✓					
A3	Record load finish time		✓					
A5	Report load start and end times and file size to Bandwidth Speedometer (2060)		✓					
A4	Compute new percent loaded		✓	✓				
A5	Select next load in progress		✓	✓				
DISPOSITION		L1	L1	L1	L1			
		Loop	Loop	Loop	Exit	Done		

Fig. 203 – ACL Check Loads Done 1980

1990		Active Content Loader Abort loads			
L1	Loop thru loads in progress	Ent	Loop	Exit	
C2	Is there a load in progress to abort?		Y	N	
A1	Select 1st load in progress	✓			
A2	Abort load in progress	✓			
A3	Mark item unloaded	✓			
A4	Select next load in progress	✓			
DISPOSITION		L1 Loop	L1 Loop	L1 Exit	Done

Fig. 204 – ACL Abort Loads 1990

2000		Active Content Loader Continue load processing			
C1	Load operation?	Prep	Stg	Load	
C2	Clip or scene?			clip	sc ene
A1	Perform prep processing (2010)	✓			
A2	Perform stage processing (2020)		✓		
A3	Perform clip load processing (2030)			✓	
A4	Perform scene load processing (2040)				✓
DISPOSITION		Done	Done	Done	Done

Fig. 205 – ACL Continue Load Processing 2000

2010	Active Content Loader Perform prep processing								
C1	Are there free load channels?	No	Yes						
A1	Allocate load channel		✓						
A2	Select Matching File (2050)		✓						
A3	Record load start time		✓						
A4	Start load of scene file		✓						
	DISPOSITION	Done	Done						

Fig. 206 – ACL Perform Prep Processing 2010

2020	Active Content Loader Perform stage processing								
C1	Are there free load channels?	No	Yes						
C2	Are there more data blocks to load?			Yes	No				
A1	Allocate load channel		✓						
A2	Pick next data block		✓						
A3	Select Matching File (2050)		✓						
A4	Record load start time		✓						
A5	Start load of data block file		✓						
	DISPOSITION	Done	C2	C1	Done				

Fig. 207 – ACL Perform Stage Processing 2020

2022	Active Content Loader Perform stage processing							
C1	Are there any objects in the scene?	No	Yes					
L2	Loop through data blocks	Ent		Loop	Exit			
C3	More data blocks needed for this scene?		Yes	No				
A1	Dup master data block			✓				
A2	Init new data block for object			✓				
A3	Copy object parameters (2025)			✓				
A4	Check Scene Checksums (2028)					✓		
	D I S P O S I T I O N	Done	L2 Loop	L2 Loop	L2 Exit	Done		

Fig. 208 - ACL Perform Stage Processing 2022

2025	Active Content Loader Copy Object Parameters							
L1	Loop through object parameters	Ent		Loop		Exit		
C2	More integer parameters?		Yes	No				
C3	More string parameters?			Yes	No			
C4	More floating point parameters?				Yes	No		
A1	Copy parameter to data block		✓	✓	✓			
A2	Add parameter to object's integer checksum		✓					
A2	Add parameter to object's string checksum			✓				
A2	Add parameter to object's float checksum				✓			
A5	Save the three checksums in the data block					✓		
	D I S P O S I T I O N	L1 Loop	L1 Loop	L1 Loop	L1 Loop	L1 Exit	Done	

Fig. 209 - ACL Copy Object Parameters 2025

2028		Active Content Loader Check Scene Checksums						
C1	Are there any objects in the scene?	No	Yes					
L2	Loop through data blocks	Ent		Loop	Exit			
C3	More data blocks for this scene?		Yes	No				
C4	Do check values match scene checksums?				Yes	No		
A1	Init integer, string, and float checksum check values		✓					
A2	Add object integer checksum to integer check value			✓				
A3	Add object string checksum to string check value			✓				
A4	Add object float checksum to float check value			✓				
A5	Display error message dialog box					✓		
A6	Halt processing (project cannot be viewed)					✓		
D I S P O S I T I O N				Done	L2	L2	L2	
				Loop	Loop	Exit	Done	Done

Fig. 210 - ACL Check Scene Checksums 2028

2030		Active Content Loader Perform clip load processing						
C1	Are there free load channels?	No	Yes					
A1	Allocate load channel		✓					
A2	Select Matching File (2050)		✓					
A3	Record load start time		✓					
A4	Start load of clip file		✓					
DISPOSITION				Done	Done			

Fig. 211 – ACL Perform Clip Load Processing 2030

2040	Active Content Loader Perform scene load processing				
C1	Are there free load channels?	No	Yes		
C2	Are there more component files to load?			Yes	No
A1	Allocate load channel		✓		
A2	Pick next component file		✓		
A3	Select Matching File (2050)		✓		
A4	Record load start time		✓		
A5	Start load of component file		✓		
DISPOSITION		Done	C2	C1	Done

Fig. 212 – ACL Perform Scene Load Processing 2040

2050	Active Content Loader Select Matching File				
C1	Is there more than one possible file?	No	Yes		
C2	Is there an exact bandwidth and cpu match?	Yes	No		
C3	Are there files with lower bandwidth and lower cpu?		Yes	No	
C4	Are there files with lower cpu and higher bandwidth			Yes	No
A1	Select the only file	✓			
A2	Select exact match file		✓		
A3	Select file with largest bandwidth \leq to actual bandwidth and with largest cpu requirement \leq to actual cpu capability			✓	
A4	Select file with smallest bandwidth \geq actual bandwidth and with largest cpu requirement \leq actual cpu capability				✓
A5	Compute distances of file requirements from actual capabilities				✓
A6	Select file with smallest overall distance				✓
DISPOSITION		Done	Done	Done	Done

Fig. 213 – ACL Select Matching File 2050

2060	Active Content Loader Report Load to Bandwidth Speedometer						
C1	Entry?	Yes					
A1	Compute load time	✓					
A2	Compute bytes per second for this load	✓					
A3	Compute running average of all loads	✓					
A4	Set bandwidth selector	✓					
	DISPOSITION	Done					

Fig. 214 – ACL Report Load to Bandwidth Speedo 2060

2070	Active Content Loader CPU Speedometer						
C1	Is it time to measure the CPU again?	Yes	No				
A1	Execute timed loop	✓					
A2	Compute instructions per second	✓					
A3	Compute running average of CPU speed	✓					
A4	Use CPU speed and frame rate to set CPU selector	✓					
	DISPOSITION	C1	C1				

Fig. 215 – ACL CPU Speedometer 2070

2080	Active Content Loader Frame Rate Speedometer							
C1	Is it time to measure the frame rate again? Yes	No						
A1	Execute timed frame loop	✓						
A2	Compute frames per second	✓						
A3	Compute running average of frame rate	✓						
A4	Use CPU speed and frame rate to set CPU selector	✓						
	DISPOSITION	C1	C1					

Fig. 216 – ACL Frame Rate Speedometer 2080